Generalized Bin Packing Problems

POLITECNICO DI TORINO

SCUOLA INTERPOLITECNICA DI DOTTORATO

Doctoral Program in Computer and Control Engineering

Final Dissertation

Generalized Bin Packing Problems

Mauro Maria Baldi

Tutor
prof. R. Tadei

Co-ordinator of the Research Doctorate Course
prof. P. Laface

February 2013

Contents

List of Tables

List of Figures

Chapter 1

Introduction

Packing problems make up a fundamental topic of combinatorial optimization. Their importance is confirmed both by their wide range of scientific and technological applications they are able to address and by their theoretical implications. In fact, they are exploited in many fields such as computer science and technologies [Ullman, 1971, Francis, 1993, Hutton, 1993], industrial applications [Boutevin et al., 2003, Freire Beirão, 2009], transportation and logistics [Akkas, 2004, Cochran and Ramanujam, 2006, Epstein, 2009, Baldi et al., 2012c], and telecommunications [Huang and Zhuang, 2000, Skorin-Kapov, 2007, Detti et al., 2009]. From a theoretical perspective, packing problems often appear as sub-problems in order to iteratively solve bigger problems [Naddef and Rinaldi, 2001, Fukunaga and Korf, 2007].

Roughly speaking, they consist in loading a set of items into proper bins in order to optimize a given objective function. Fundamentally, packing problems can be classified into two families: the one of bin packing problems and the one of knapsack problems. These two families are very different in terms of formulations, objective functions, methodologies, and, above all, items nature. In fact, whilst in bin packing problems items are compulsory (i.e., they all must be loaded into bins), in knapsack problems they are non-compulsory.

Although packing problems play a fundamental role in all the aforementioned settings, there is a gap in terms of comprehensive study in the literature. In fact, the joint presence of both compulsory and non-compulsory items has not been considered yet. This particular setting arises in many real-life applications, not yet

addressed or only partially addressed by the current state-of-the-art packing problems. Furthermore, little has been done in terms of unified methodologies, and different techniques have been used in order to solve packing problems with different objective functions. In particular, none of these techniques is able to address the presence of compulsory and non-compulsory items at the same time.

In order to overcome a noteworthy portion of this gap, we formulated a new packing problem, named the **Generalized Bin Packing Problem** (**GBPP**), characterized by both compulsory and non-compulsory items, and multiple item and bin attributes.

Packing problems have also been studied within stochastic settings where the items are affected by uncertainty. In these settings, there are fundamentally two kinds of stochasticity concerning the items: 1) stochasticity of the item attributes, where one attribute is affected by uncertainty and modeled as a random variable or 2) stochasticity of the item availability, i.e., the items are not known a priori but they arrive on-line in an unpredictable way to a decision maker.

Although packing problems have been studied according to these stochastic variants, the **GBPP** with uncertainty on the items is still an open problem. Moreover, to the best of our knowledge, also the on-line variant of the **VCSBPP**, the most similar problem to the **GBPP**, has not been studied yet. Therefore, we have also studied two stochastic variants of the **GBPP**, named the **Stochastic Generalized Bin Packing Problem** (**S-GBPP**) and the **On-line Generalized Bin Packing Problem** (**OGBPP**), and the on-line variant of the **Variable Cost and Size Bin Packing Problem** (**VCSBPP**), the **On-line Variable Cost and Size Bin Packing Problem** (**OVCSBPP**).

Our main results concern the development of models and unified methodologies of these new packing problems, making up, as done for the **Vehicle Routing Problem** (**VRP**) with the definition of the so called **Rich Vehicle Routing Problems**, a new family of advanced packing problems named **Generalized Bin Packing Problems**.

In the **GBPP**, a set of items, characterized by volume and profit, and a set of bins, characterized by capacity and cost, are given. The item set is split into two subsets: the subset of compulsory items (which must be loaded into bins) and the subset of non-compulsory items (which may not be loaded). The bins are classified

by types, such that bins belonging to the same type have the same capacity and cost. The goal is to properly select a subset of profitable non-compulsory items to be loaded together with the compulsory ones into the appropriate bins in order to minimize the total net cost and satisfying capacity and bin usage constraints. The total net cost is given by the difference between the total cost of the selected bins and the total profit of the loaded non-compulsory items.

The **GBPP** yields a relevant amount of contributions. The most relevant contribution is the presence of both compulsory and non-compulsory items and of multiple item and bin attributes. This innovative feature allows us to address and collect several bin packing and knapsack problems at the same time into a unique structure. The **GBPP**, indeed, is able to gather the following packing problems: the **Bin Packing Problem** (**BPP**), the **Variable Sized Bin Packing Problem** (**VSBPP**), the **VCSBPP**, the **Knapsack Problem** (**KP**), the **Multiple Knapsack Problem** (**MKP**), and the **Multiple Knapsack Problem with identical capacities** (**MKPI**). Moreover, the **GBPP** is able to address new applications. In logistics, where changes arising in the supply chain and fleet management due to cross-continental fleet flows and multi-modal and green logistics have forced researchers and practitioners to redefine their processes [Cohn and Barnhart, 1998, Shintani et al., 2007]. The **GBPP** is a contribution in this direction, as it defines a packing problem able to simultaneously consider bin costs and item profits and takes into account restrictions on the bin availability and their heterogeneity in terms of cost and volume. The **GBPP** brings also innovation in the area of airfreight transportation, where items are loaded according to their volume [Li, 2011]. Here, the **GBPP** is able to describe the fundamental role played by the trade-off between shipping costs and item profits, which arises in many transportation settings. The **GBPP** also brings contributions in the waste collection problem at a tactical level. In the literature, this problem is tackled at the operational level, where the routes are determined solving a **VRP** [Bianchi de Aguiar, 2010, Bianchi de Aguiar et al., 2012]. Given the demand of ordinary waste and hazardous and bulky waste (which yield a profit to the waste company), the **GBPP** determines proper vehicles (represented by bins) in order to fulfill an optimized picking. Afterward, routes are determined at the operational level, solving the **VRP**.

We give two formulations of the **GBPP** and we propose variegated methods in

terms of quality and computational time.

Furthermore, the availability of these methodologies for the **GBPP** yields the great flexibility of using the same techniques to address different problems, avoiding to change the solution methods every time the problem changes.

The **Stochastic Generalized Bin Packing Problem** is a further generalization of the **GBPP** where item profits are no longer fixed, but depend on bins where items will be loaded. Moreover, they are random variables with unknown probability distribution. The goal is to maximize the expected total net profit, given by the difference between the expected total profit of the loaded items and the total cost of the used bins, while satisfying volume and bin availability constraints.

The **S-GBPP** is able to address a more general setting where each item profit depends on the bin where the item is loaded and it is described by a random variable. This generalization allows us to address new applications, in particular in logistics, where the freight consolidation is essential to optimize the delivery process. Profit random terms represent a series of handling operations for bin loading that must be performed at the logistic platforms, and these operations could significantly affect the final total profit of the loading [Tadei et al., 2002].

Moreover, the **S-GBPP** is able to address the **Railway Track Maintenance Planning Problem**, where maintenance operations (named warnings) must be scheduled into time-slots and their costs are uncertain and depend on the time-slots where they are assigned [Heinicke et al., 2012, 2013].

For this problem, we give a stochastic model and, applying the extreme value theory, we derive a deterministic approximation.

In the **On-line Generalized Bin Packing Problem**, the items arrive on-line to a decision maker, therefore no information on the items is known a priori. Only when an item has been received, its information is revealed.

This problem arises in all those applications where orders arrive on-line. This is the case, for instance, of freight forwarders: specialized companies arranging shipments between logistics providers and customers, and playing the role of intermediary between the involved parts.

We study a wide range of algorithms in order to test whether the available tools in the literature (i.e., the asymptotic and absolute worst case ratios) are still effective when a richer setting as the **OGBPP** is tackled. Our study reveals a stronger

result than the one achieved in the **On-line Knapsack Problem**(**OKP**). In fact, as Iwama and Zhang [2007, 2010] showed that the **OKP** is not competitive (i.e., its absolute worst case ratio is infinite), we prove that, for the proposed algorithms, it is even impossible to apply the definition of these tools. This behavior occurs also in the **On-line Generalized Bin Packing Problem with item profits proportional to item volumes** (**OGBPP**$_\kappa$), the on-line variant of the **Generalized Bin Packing Problem with item profits proportional to item volumes** (**GBPP**$_\kappa$), a particular case of the **GBPP** where item profits are proportional to their corresponding volumes through a positive coefficient κ. This particular problem arises in many real-life applications.

We believe that the ultimate packing problem for which it is possible to compute a performance ratio is the **On-line Variable Cost and Size Bin Packing Problem**, the closest problem to the **GBPP**, where items arrive on-line, but still without the presence of non-compulsory ones. As for the **OGBPP**, this problem arises in many settings where orders arrive on-line.

For this problem, we could generalize the work of Li and Chen [2006] to a more general setting, still guaranteeing the same performance ratio.

This thesis is organized as follows. In Chapter 2, we present a detailed state of the art of the **GBPP** and of the packing problems it is able to address.

In Chapter 3, we present the **GBPP**. After introducing the problem, we give two models and propose bounds. In particular, we propose two lower bounds, one solving an **Aggregate Knapsack Problem** (**AKP**) and a second one computed in terms of column generation. Then, we show how, basing on these lower bounds, we can also compute accurate upper bounds. We also compute upper bounds in terms of constructive heuristics. Afterwards, we present new instance sets for tackling the problem and extensive computational results.

In Chapter 4, we give an exact method, named branch-and-price, for solving the **GBPP**. Our method, rooted on a previous work of Bettinelli et al. [2010], consists in a two-layer branching strategy. We also propose an approximate method, named beam search, which is based on the branch-and-price architecture. Extensive computational results are also presented.

In Chapter 5, we present the **S-GBPP**. We first provide a more general deterministic model, where the profit of each non-compulsory item depends on the bin

where it is loaded. Then, we present the stochastic model, where each profit becomes a random variable with unknown distribution. Starting from this model and applying the extreme value theory, we derive a deterministic approximation of the **S-GBPP**.

In Chapter 6, we present the **OGBPP**, the **OGBPP**$_{\kappa}$, and the **OVCSBPP**, and we propose and study a wide range of algorithms addressing these problems.

Conclusions and future developments of the research activity are reported in Chapter 7.

Chapter 2

Literature review

In this chapter, we recall the literature of the GBPP and of its related problems. These are: the BPP, the VSBPP, the VCSBPP, the KP, the MKP, and the MKPI.

The GBPP is a novel packing problem recently introduced by Baldi et al. [2012a]. In their paper, the authors propose two models and preliminary bounds. A branch-and-price method and beam search heuristics have been proposed in [Baldi et al., 2012b]. The stochastic variant of the problem has been studied by Perboli et al. [2012].

The most classical bin packing problem addressed by the GBPP is the BPP. The BPP is the simplest mono-dimensional bin packing problem, introduced by Ullman [1971], which consists in finding the minimum number of bins (all having the same capacity) in order to accommodate a set of items satisfying capacity constraints. A noteworthy pioneering work has been conducted by Johnson who, in his papers and PhD thesis, proposed and studied preliminary algorithms, some of them still in the vanguard. In particular, in [Johnson, 1973a], he proposed the Next Fit (NF) algorithm and proved that its performance ratio is 2. In [Johnson et al., 1974], he proposed the First Fit (FF), Best Fit (BF), First Fit Decreasing (FFD), and Best Fit Decreasing (BFD) algorithms and showed that their performance ratios are, 17/10 for FF and BF and 11/9 for FFD and BFD. FFD and BFD are still very used nowadays, sometimes combined with local improvement heuristics [Schwerin and Wäscher, 1997]. Basing their studies on the work of Johnson,

several authors proposed improved algorithms in the years. Yao [1980] presented the REFINED FIRST FIT algorithm, with performance ratio 5/3, and proved that any on-line algorithm must have a performance ratio of at least 3/2. Afterward, van Vliet [1992] increased the lower bound to 1.54014. Lee and Lee [1985] presented a family of bounded space algorithms, named HARMONIC$_M$, with performance ratio approaching $h_\infty \approx 1.6910$ from above as $M \to +\infty$, and showed that h_∞ is even a lower bound for this class of problems. The authors also presented the REFINED HARMONIC algorithm, with performance ratio $373/228 \approx 1.63597$. To the best of our knowledge the best result to date is due to Seiden [2002] who proposed the HARMONIC++ algorithm with performance ratio at most 1.58889.

Preliminary bounds to the BPP were proposed by Martello and Toth [1990]. New lower bounds were developed by Fekete and Schepers [2001] by means of dual feasible functions. On the basis of this paper, Crainic et al. [2007] developed fast and more accurate lower bounds, able to reduce the optimality gap for a number of hard instances. A different approach was defined in [Vanderbeck, 1996], where the author proposed a formulation with an exponential number of variables and a column generation lower bound procedure for the Bin Packing and the Cutting Stock problems. Several heuristics were also proposed [Martello and Toth, 1990], e.g., the polynomial-time approximation schemes of de la Vega and Lueker [1981] and Karmarkar and Karp [1982] allowing to approximate an optimal solution within $1 + \epsilon$, for any fixed $\epsilon > 0$. However, these results are difficult to apply in practice, due to the enormous size of the constants characterizing the polynomials.

Han et al. [2010] studied a variant of the problem where items have also arrival and departure time. Both the off-line and the on-line versions of the problem were studied, with particular attention to the case of unit fraction items, i.e., when the sizes of the items are at most $1/i$, for some integer i.

The BPP has also been studied with respect to less standard ratios. Epstein and van Stee [2005] studied the problem by means of resource augmentation. Kouakou et al. [2005] studied the problem with respect to the differential competitivity ratio. Finally, György et al. [2010] studied a much more restrictive variant of the problem with just one open bin and the decision maker can only decide whether to pack the next item into the open bin or close the incumbent open bin and load the item into a new bin (which, of course, becomes the new open bin). If the decision maker decides

not to close the current open bin and the next item does not fit into it, then the item is lost. The goal is to minimize the wasted space plus the number of lost items. This version of the problem is called on-line Sequential Bin Packing Problem.

Another variant of the BPP was studied by Li and Chen [2006], where the bins have all the same capacity but they are also characterized by a non-decreasing concave cost function. The authors proved that, for this variant of the problem, FF and BF heuristics have absolute worst case ratio equal to 2, whilst FFD and BFD have absolute worst case ratio equal to 1.5. For this problem, Leung and Li [2008] developed a polynomial time approximation algorithm such that, for any positive ϵ, the asymptotic worst case ratio is $1 + \epsilon$. Epstein and Levin [2012] have recently designed an asymptotic fully polynomial time approximation scheme for this problem. In their work, the authors have also proposed a fast approximation algorithm with asymptotic worst case ratio of 1.5.

The stochastic variant of the BPP has been studied by Coffman Jr. et al. [1980], Lueker [1983], Rhee and Talagrand [1993a,b]. In these papers, the source of uncertainty is the item volume and strong hypotheses on the probability distribution of the random terms are usually done. Recently, Peng and Zhang [2012] have studied a more general stochastic variant, where both item volumes and bin capacities are uncertain.

Another problem addressed by the GBPP is the VSBPP, where bins with different sizes are available and the goal is to minimize the wasted space. This problem was first investigated by Friesen and Langston [1986]. The authors provided one on-line and two off-line algorithms and proved that their worst case ratio is respectively 2, 3/2, and 4/3. Murgolo [1987] presented an approximation scheme which, for any positive ϵ, produces a scheme with performance ratio $1+\epsilon$. Moreover, his algorithm is polynomial even with respect to $1/\epsilon$. Chu and La [2001] proposed four greedy approximation algorithms with absolute worst case ratio respectively equal to 2, 2, 3, and $2 + \ln 2$. The authors also showed that these bounds are tight. Seiden [2000] proposed an optimal on-line algorithm for the bounded space (i.e., the number of open bins is constant) problem. Seiden et al. [2003] proposed improved bounds but with two bin sizes only. Zha provided a lower bound of 2.245 with respect to the absolute worst case ratio and proposed a simple on-line algorithm with absolute worst case ratio equal to 3. The author also studied the problem from

another point of view: if one were allowed to design k bin sizes, what sizes should be chosen such that, for any instance, the wasted space is minimized?

Monaci [2002] presented a series of lower bounds and solution methods (both heuristic and exact) for the VSBPP. The author also introduced instance sets for the problem considering up to 500 items. His exact method was able to solve most instances to optimality.

The VSBPP can also be seen as a special case of the Multiple Length Cutting Stock Problem(MLCSP). In this problem, the item demand can be more than one and different types of stocks (which are equivalent to the bins) are available. Exact methods for the MLCSP have been proposed by Belov and Scheithauer [2002]. Alves and Valério de Carvalho [2007] first proposed an improved column generation technique trying to solve the VSBPP to optimality. One year later, the same authors introduced a branch-and-cut-and-price algorithm for the MLCSP [Alves and Valério de Carvalho, 2008].

An on-line variant of the VSBPP was introduced by Zhang [1997] where items are known and this time are the bins to arrive on-line, one by one. The author proved that, for this problem, both NF and FFD algorithms have performance ratio 2.

The problem studied by Zhang [1997] was also the starting point for the work of Epstein et al. [2011] who studied the on-line Variable Sized Bin Packing Problem with conflicts.

The VCSBPP is a generalization of the VSBPP, where all items must be loaded, but bins can be chosen among several types differing in volume and cost. The total accommodation cost, computed as the total cost of the used bins, must be minimized. A number of studies have been dedicated to the VCSBPP. Kang and Park [2003] studied the problem assuming that the cost of the unit size of each bin does not increase as the bin size increases. The authors proposed two greedy algorithms and computed their asymptotic worst case ratio under three assumptions: 1) the sizes of items and bins are divisible (i.e., the succeeding item (bin) exactly divides the previous item (bin)), 2) the sizes of bins are divisible, and 3) the sizes of bins are not divisible. The authors proved that both algorithms yield an asymptotic worst case ratio equal to 1 (i.e., the two algorithms are optimal) under assumption 1, equal to 11/9 under assumption 2, and equal to 3/2 under assumption 3. For

this problem, Epstein and Levin [2008] designed an asymptotic polynomial time approximation scheme. Correia et al. [2008] proposed a formulation that explicitly includes the bin volumes occupied by the corresponding packings, together with a series of valid inequalities improving the quality of the lower bounds obtained from the linear relaxation of the proposed model. The authors also introduced a large set of instances with up to 1000 items and used them to analyze the quality of the lower bounds. Crainic et al. [2011] proposed tight lower and upper bounds, which can be computed within a very limited computing time, and were able to solve to optimality all the instances proposed in [Correia et al., 2008]. The authors also presented a first computational study of the sensitivity of the optimal cost with respect to the cost definition [Crainic et al., 2011]. Approximation algorithms have been proposed by Haouari and Serairi [2009] and Hemmelmayr et al. [2012]. Recently, Bettinelli et al. [2010] introduced a branch-and-price algorithm for the resolution of a variant of the VCSBPP with the addition of filling constraints. These constraints imply that, due to stability reasons within the bins, each bin must be filled at least at a minimum security level. To the best of our knowledge, the latest work dealing with exact methods for solving the VCSBPP is due to Haouari and Serairi [2011], in which the authors proposed lower bounds and an exact branch-and-bound algorithm. Fazi et al. [2012] have recently studied the Stochastic VCSBPP, with the addition of time constraints.

The GBPP is also able to generalize three knapsack problems: the KP, the MKP, and the MKPI. The KP is a deeply studied Combinatorial Optimization problem where, given a set of items characterized by volume and profit, the goal is to find a proper subset such that the profit is maximized and the sum of volumes of the selected items does not exceed the capacity of the only available bin, namely the knapsack. The MKP is a generalization of the KP due to the presence of more than one knapsack. Finally, the MKPI is a particular case of the MKP, where all the knapsacks have the same capacity. All these problems are thoroughly discussed in [Martello and Toth, 1990, Pisinger, 1995, Kellerer et al., 2004].

The OKP has been studied by Iwama and Taketomi [2002], Iwama and Zhang [2007, 2010]. In these papers, the authors prove that, for this problem, the worst case ratio is infinite. For this reason, they also study the on-line variant with removable items and resource augmentation and, for this variant of the problem, the proposed

algorithms show finite competitive ratios.

In the stochastic variant of the problem, named the Stochastic Knapsack Problem (SKP), the source of uncertainty is usually the item profit, and strong hypotheses on its probability distribution are made [Goel and Indyk, 1999, Ross and Tsang, 1989]. Some papers also consider the item volume stochasticity. Dean et al. [2008] present some policies to decide whether to load the items into the knapsack, showing how an adaptive loading policy outperforms a non-adaptive one. In Kosuch and Lisser [forthcoming], a variant of the SKP with normally distributed volumes is presented. The authors derive a two-stage SKP where, contrary to the single-stage SKP, items can be added to or removed from the knapsack at the moment the actual volumes become known (second stage).

Chapter 3

The Generalized Bin Packing Problem: models and bounds

In this chapter, we introduce the GBPP together with models and bounds. After defining the problem in a formal way, we present two mixed integer programming formulations. The first is based on item-to-bin assignment decisions and requires a polynomial number of variables and constraints. This formulation shows how the GBPP generalizes several packing problems, but is not computationally efficient. It is, however, the starting point for computing the first lower bound. Moreover, it is also the starting point for the S-GBPP, introduced in Chapter 5. We, thus, introduce a second model, based on feasible loading patterns and set covering ideas. Despite requiring an exponential number of variables, this formulation is much more efficient. We then present several procedures in order to compute lower and upper bounds to the GBPP and show their accuracy and efficiency through extensive computational experiments. A large number of instance sets for the GBPP are introduced. The instance sets are designed to challenge the proposed procedures and thus provide insight into the impact of different parameters on the optima.

This chapter is organized as follows. The two GBPP formulations are introduced in Section 3.1; lower and upper bounds are presented in Sections 3.2 and 3.3, respectively. Instance sets and computational results are presented and discussed in Section 3.4.

3.1 Problem Definition and Formulation

The GBPP considers a set of items characterized by volume and profit and sets of bins of various types characterized by volume and cost. A subset of the items which we call *compulsory* must be loaded, while a selection has to be made among the *non-compulsory* ones. The objective is to select the non-compulsory items to load and the bins into which to load the compulsory and the selected non-compulsory items in order to minimize the total net cost. This is given by the difference between the total cost of the used bins and the total profit of the loaded items.

In this section, we present a formal description of the GBPP and we introduce two formulations of the problem. The first formulation extends to the GBPP the assignment model of the BPP [Martello and Toth, 1990]. Although this kind of formulation is not often used in practice, we exploit it to derive a first lower bound. Then, we consider a set covering formulation of the problem, which is the starting point for a column generation procedure, which allows us to derive a second lower bound and upper bounds as well.

3.1.1 Notation

Let $\mathcal{I}$ denote the set of items and w_i and p_i be the volume and profit of item $i \in \mathcal{I}$. Define $\mathcal{I}^{\mathrm{C}} \subseteq \mathcal{I}$ the subset of compulsory items and $\mathcal{I}^{\mathrm{NC}} = \mathcal{I} \setminus \mathcal{I}^{\mathrm{C}}$ the subset of non-compulsory items. Let $\mathcal{J}$ denote the set of available bins and $\mathcal{T}$ be the set of bin types. For any bin $j \in \mathcal{J}$, let $\sigma(j) = t \in \mathcal{T}$ be the type t of bin j. Define, for each bin type $t \in \mathcal{T}$, the minimum L_t and the maximum U_t number of bins of that type that may be selected, as well as the cost C_t and the volume W_t of the bin. Finally, denote $U \leq \sum_{t \in \mathcal{T}} U_t$ the total number of available bins of any type.

The item-to-bin accommodation rules of the GBPP are stated as follows

- All items in I^{C} must be loaded
- For all used bins, the sum of the volumes of the items loaded into a bin must be less than or equal to the bin volume
- The number of bins used for each type $t \in \mathcal{T}$ must be within the lower and upper availability limits L_t and U_t

- The total number of used bins cannot exceed the total number of available bins U.

Infeasibility may arise when the available bins are not sufficient to load all compulsory items. To address this issue, we add a special bin v of volume $W_v = \sum_{i \in \mathcal{I}^{\mathrm{C}}} w_i$, thus able to load all compulsory items, and set its cost C_v to a value much higher than the costs of the remaining bins in order to discourage its use, e.g., $C_v \gg \sum_{t \in \mathcal{T}} C_t$.

3.1.2 Assignment formulation of the GBPP

Consider the following decision variables:

- Bin selection binary variables y_j equal to 1 if bin $j \in \mathcal{J}$ is used, 0 otherwise
- Item-to-bin assignment binary variables x_{ij} equal to 1 if item $i \in \mathcal{I}$ is loaded into bin $j \in \mathcal{J}$, 0 otherwise.

An assignment model of the GBPP can then be formulated as follows

$$\min \quad \sum_{j \in \mathcal{J}} C_j y_j - \sum_{j \in \mathcal{J}} \sum_{i \in \mathcal{I}^{\mathrm{NC}}} p_i x_{ij} \tag{3.1}$$

$$\text{subject to} \quad \sum_{i \in \mathcal{I}} w_i x_{ij} \leq W_j y_j \qquad j \in \mathcal{J} \tag{3.2}$$

$$\sum_{j \in \mathcal{J}} x_{ij} = 1 \qquad i \in \mathcal{I}^{\mathrm{C}} \tag{3.3}$$

$$\sum_{j \in \mathcal{J}} x_{ij} \leq 1 \qquad i \in \mathcal{I}^{\mathrm{NC}} \tag{3.4}$$

$$\sum_{j \in \mathcal{J}: \sigma(j) = t} y_j \leq U_t \qquad t \in \mathcal{T} \tag{3.5}$$

$$\sum_{j \in J: \sigma(j) = t} y_j \geq L_t \qquad t \in \mathcal{T} \tag{3.6}$$

$$\sum_{j \in \mathcal{J}} y_j \leq U \tag{3.7}$$

$$y_j \in \{0, 1\} \qquad j \in \mathcal{J} \tag{3.8}$$

$$x_{ij} \in \{0, 1\} \qquad i \in \mathcal{I}, \quad j \in \mathcal{J}. \tag{3.9}$$

The objective function (3.1) minimizes the total net cost of the packing, given by the difference between the total cost of the used bins and the total profit of the selected non-compulsory items. The profit of the compulsory items is not considered in the objective function because it is a constant. Regarding the type of optimization, we choose to present the minimization version to follow the tradition of bin packing problems. The equivalent formulation obtained by maximizing the total net profit (total profit minus total cost) would recall the knapsack problem setting.

Constraints (3.2) have the double effect of linking the usage of bins to the accommodation of items and bounding the capacity of each used bin. Constraints (3.3) and (3.4) ensure that each compulsory and not-compulsory item is loaded into exactly one and at most one bin, respectively. Constraints (3.5) and (3.6) enforce the maximum and minimum number of available bins of each type, while Constraint (3.7) limits the total number of selected bins. Constraints (3.8) and (3.9) enforce the integrality nature of the decision variables. Notice that, Constraints (3.5) and (3.6) could be implicitly managed, the former by limiting the number of y_j variables to U_t for type t (i.e., defining the appropriate number of bins only), and the latter by setting $y_j = 1$ for $j = 1, \ldots, L_t$. We prefer to keep the constraints in the formulation, however, for consistency with the set covering model of Section 3.1.4.

The assignment model (3.1)-(3.9) is named *AM* and its continuous relaxation *R-AM*. *AM* involves a polynomial number of variables and constraints. It is not suitable for developing efficient algorithms, however, due to the significant solution-space symmetry of the item-to-bin assignment variables, which is typical of these packing models. Yet, as mentioned above, *AM* is the starting point to compute our first lower bound named LB_1 and to formulate the S-GBPP in Chapter 5. Furthermore, *AM* is also suitable to show how the GBPP generalizes some classic bin packing and knapsack problems. This issue is addressed next.

3.1.3 Generalization of classic bin packing and knapsack problems

The assignment formulation (3.1)-(3.9) is useful to show how the GBPP can generalize a number of classic packing problems. As mentioned in Chapters 1 and 2, the GBPP is able to address the BPP, the VSBPP, the VCSBPP, the KP, the

MKP, and the MKPI. The BPP can be modeled by considering a single bin type with $C_j = 1, j \in \mathcal{J}$ and $\mathcal{I}^{\text{NC}} = \emptyset$, i.e., all items must be loaded. Constraints (3.4) - (3.6) are then redundant and the objective function becomes the minimization of the number of bins, which is characteristic of the BPP.

Allowing several bin types and $\mathcal{I}^{\text{NC}} = \emptyset$ yields the VCSBPP, where the total cost of the selected bins $\sum_{j \in \mathcal{J}} C_j y_j$ is minimized (Constraints (3.4) become redundant) [Monaci, 2002, Crainic et al., 2011]. Notice that, an equivalent formulation for the VCSBPP can be obtained by imposing $\mathcal{I}^{\text{C}} = \emptyset$ and setting the item profit higher than the cost of any bin type, $p_i > \max_{j \in \mathcal{J}} C_j$, which makes any bin profitable even when only one item is loaded into it.

The VSBPP can also be addressed setting $\mathcal{I}^{\text{NC}} = \emptyset$ and $C_j = W_j, \forall j \in \mathcal{J}$.

The GBPP can similarly generalize a number of knapsack problems. Specifically, the GBPP reduces to the KP by setting $|\mathcal{T}| = 1$, $|\mathcal{J}| = 1$, and $\mathcal{I}^{\text{C}} = \emptyset$. The MKP can be modeled by setting $|\mathcal{T}| > 1$, $|\mathcal{J}| = m$, where m is the number of knapsacks, and $\mathcal{I}^{\text{C}} = \emptyset$. Finally, the MKPI is addressed by setting $|\mathcal{T}| = 1$, $|\mathcal{J}| = m$, and $\mathcal{I}^{\text{C}} = \emptyset$.

3.1.4 Set Covering formulation of the GBPP

We introduce a set covering formulation of the GBPP based on feasible loading patterns for the bins.

A *feasible loading pattern* k_t for a bin of type $t \in \mathcal{T}$ is a set of items that may be loaded into the bin while satisfying all dimension restrictions and accommodation rules. Let $\mathcal{K}_t = \{k_t\}$ be the set of all feasible loading patterns for bin type $t \in \mathcal{T}$. A feasible loading pattern k_t is described by a vector $\boldsymbol{a_{k_t}}$ of indicator functions $a^i_{k_t}$, $i \in \mathcal{I}$, $k_t \in \mathcal{K}_t$, $t \in \mathcal{T}$, such that $a^i_{k_t} = 1$ if item i is packed into pattern k_t, 0 otherwise. The cost of pattern k_t is then the difference between the cost of the bin type t and the total profit of non-compulsory items in that pattern:

$$c_{k_t} = C_t - \sum_{i \in \mathcal{I}^{\text{NC}}} p_i a^i_{k_t}. \tag{3.10}$$

We define the bin loading pattern selection variables λ_{k_t} equal to 1 if pattern $k_t \in \mathcal{K}_t$ is used, 0 otherwise. The set covering formulation of the GBPP may then

be written as follows

$$\min \quad \sum_{t\in\mathcal{T}}\sum_{k\in\mathcal{K}_t} c_{k_t}\lambda_{k_t} \tag{3.11}$$

$$\text{subject to} \quad \sum_{t\in\mathcal{T}}\sum_{k\in\mathcal{K}_t} a^i_{k_t}\lambda_{k_t} = 1 \quad i\in\mathcal{I}^{\mathrm{C}} \ \text{(dual variable } \mu_i \text{ free)} \tag{3.12}$$

$$\sum_{t\in\mathcal{T}}\sum_{k\in\mathcal{K}_t} a^i_{k_t}\lambda_{k_t} \le 1 \quad i\in\mathcal{I}^{\mathrm{NC}} \ \text{(dual variable } \nu_i \le 0) \tag{3.13}$$

$$\sum_{k\in\mathcal{K}_t} \lambda_{k_t} \le U_t \quad t\in\mathcal{T} \qquad \text{(dual variable } \alpha_t \le 0) \tag{3.14}$$

$$\sum_{k\in\mathcal{K}_t} \lambda_{k_t} \ge L_t \quad t\in\mathcal{T} \qquad \text{(dual variable } \beta_t \ge 0) \tag{3.15}$$

$$\sum_{t\in\mathcal{T}}\sum_{k\in\mathcal{K}_t} \lambda_{k_t} \le U \qquad \text{(dual variable } \epsilon \le 0) \tag{3.16}$$

$$\lambda_{k_t} \in \{0,1\} \qquad k_t\in\mathcal{K}_t,\ t\in\mathcal{T}. \tag{3.17}$$

The objective function (3.11) minimizes the total cost of the selected bin loading patterns.

Since the feasibility of the item-to-bin accommodation is guaranteed by the feasibility of the loading patterns, the constraints equivalent to (3.2) in the model *AM* are not required anymore. Constraints (3.12)-(3.16) have the same meaning as (3.3)-(3.7), and (3.17) are the integrality constraints.

The set covering model (3.11)-(3.17) is named *SC* and its continuous relaxation *R-SC*. *SC*, when compared to *AM*, has the advantage to separate the feasibility phase from the optimality one. The feasibility phase is already addressed by the pattern generation, whilst the model is only devoted to find an optimal combination of patterns.

Whilst, of course, models *AM* and *SC* have the same optimum, the optima of *R-AM* and *R-SC* are generally different. In the following, we prove that the lower bound to the GBPP obtained by optimizing *R-SC* is not weaker than that obtained by optimizing *R-AM*.

Property 1. *Given any solution* x_2 *of* R-SC *(which is feasible by construction), a corresponding feasible solution* $x_1(x_2)$ *of* R-AM *can be built as follows. For any* $\lambda_{k_t} = 1$ *of* x_2 *assign in* R-AM *the items of pattern* k_t *to bin* j_{k_t} *by putting* $x_{ij_{k_t}} =$

$a^i_{k_t}\lambda_{k_t}$ *in the solution* $x_1(x_2)$*. The two solutions have the same value.*

Proof. Trivial. □

Theorem 1. *Let* $L_{\text{R-AM}} = optimum(\text{R-AM})$ *and* $L_{\text{R-SC}} = optimum(\text{R-SC})$*, then* $L_{\text{R-AM}} \leq L_{\text{R-SC}}$.

Proof. By contradiction, let us suppose that an instance of *R-AM* such that $L_{R\text{-}AM} > L_{R\text{-}SC}$ there exists. Let $\overline{x}_2$ be the optimal solution associated to $L_{R\text{-}SC}$. By using Property 1, we can build a solution $\overline{x}_1(\overline{x}_2)$ of *R-AM* with value $L_{R\text{-}SC}$, which contradicts the optimality of $L_{R\text{-}AM}$. □

3.2 Lower bounds

We introduce two lower bounds that can be computed starting from each of the two problem formulations. The assignment model *AM* is the basis for a lower bound which can be calculated by solving an Aggregate Knapsack Problem (AKP). We show how to derive the AKP from the model *AM* in Section 3.2.1.

The second lower bound is derived from the set covering formulation *SC* and is calculated by applying a column generation technique where, at each step, a new feasible pattern (i.e., a new column for the restricted master problem) is found by solving a knapsack problem (see Section 3.2.2).

3.2.1 Lower bound through the Aggregate Knapsack Problem

To derive an Aggregate Knapsack Problem from the assignment model *AM* presented in Section 3.1.2, we aggregate Constraints (3.2) into a unique inequality by summing them up. We thus consider an aggregate knapsack, which may be thought of as a unique large bin with volume equal to the total volume of the bins. We have

$$\sum_{j \in \mathcal{J}} \sum_{i \in \mathcal{I}} w_i x_{ij} \leq \sum_{j \in \mathcal{J}} W_j y_j \quad \Longrightarrow \quad \sum_{i \in \mathcal{I}^C} w_i \sum_{j \in \mathcal{J}} x_{ij} + \sum_{i \in \mathcal{I}^{NC}} w_i \sum_{j \in \mathcal{J}} x_{ij} \leq \sum_{j \in \mathcal{J}} W_j y_j.$$

Note that, by (3.3), for any compulsory item i, $\sum_{j \in \mathcal{J}} x_{ij} = 1$, whilst, for any non-compulsory item i, the variable x_{ij} can be reduced to x_i, which states whether item i is put into the aggregate knapsack or not. Therefore, we have

$$\sum_{i \in \mathcal{I}^C} w_i + \sum_{i \in \mathcal{I}^{NC}} w_i x_i \leq \sum_{j \in \mathcal{J}} W_j y_j. \tag{3.18}$$

We drop Constraints (3.5) and (3.6) by implicitly managing them as indicated in Section 3.1.2. Constraint (3.7) is kept, as one cannot implicitly manage it. A first lower bound, named LB_1, can then be found by solving the following AKP

$$\begin{aligned}
\min \quad & \sum_{j \in \mathcal{J}} C_j y_j - \sum_{i \in \mathcal{I}^{\mathrm{NC}}} p_i x_i && && (3.19)\\
\text{subject to} \quad & \sum_{i \in \mathcal{I}^C} w_i + \sum_{i \in \mathcal{I}^{NC}} w_i x_i \leq \sum_{j \in \mathcal{J}} W_j y_j && && (3.20)\\
& \sum_{j \in \mathcal{J}} y_j \leq U && && (3.21)\\
& y_j \in \{0,1\} && j \in \mathcal{J} && (3.22)\\
& x_i \in \{0,1\} && i \in \mathcal{I}^{\mathrm{NC}}. && (3.23)
\end{aligned}$$

Since LB_1 comes from the resolution of the AKP, we also refer to it as an *aggregate knapsack lower bound*. Note that, when all items are compulsory, one can use bounds from the literature. In this case, (3.19)-(3.23) reduces to

$$\begin{aligned}
\min \quad & \sum_{j \in \mathcal{J}} C_j y_j && && (3.24)\\
\text{subject to} \quad & \sum_{i \in \mathcal{I}^C} w_i \leq \sum_{j \in \mathcal{J}} W_j y_j && && (3.25)\\
& \sum_{j \in \mathcal{J}} y_j \leq U && && (3.26)\\
& y_j \in \{0,1\} && j \in \mathcal{J}, && (3.27)
\end{aligned}$$

which is the model used by Crainic et al. [2011] to compute lower bounds to the VCSBPP.

3.2.2 Lower bound through column generation

This lower bound is computed from the *R-SC* model through column generation and is named LB_2. It is widely used in bin packing problems [Alves and Valério de Carvalho, 2007, Vanderbeck, 1996] and provides the means to implicitly deal with a large number of variables.

The column generation approach applied to the *R-SC* model consists in starting with a relatively small set of feasible patterns $\mathcal{P}$, which correspond to a restricted problem named $R - SC_R$. First we solve $R - SC_R$ and then attempt to generate new feasible patterns with negative reduced cost. If successful, these are added to $\mathcal{P}$ and the procedure is restarted. Otherwise, we have found an optimal solution of *R-SC* and the procedure stops with a lower bound to the GBPP.

The main steps of the procedure are as follows

1. Find an initial feasible solution of the GBPP and the corresponding set $\mathcal{P}$
2. Solve to optimality $R - SC_R$ and let L_{R-SC_R} be its optimum
3. For each bin type $t \in \mathcal{T}$
   (a) Find the pattern variable $\lambda_{\overline{k}_t}$, $\overline{k}_t \in \mathcal{K}_t$ with the smallest reduced cost $r_{\overline{k}_t}$, among all non-basic pattern variables λ_{k_t} of the optimal solution of $R - SC_R$
   (b) If $r_{\overline{k}_t} < 0$, $\mathcal{P} = \mathcal{P} \cup \{\lambda_{\overline{k}_t}\}$
4. If $r_{\overline{k}_t} \geq 0$ for all bin types t, then stop and L_{R-SC_R} is the lower bound to the GBPP, otherwise, go to 2.

Note that, in Step 3, the procedure adds at most $|\mathcal{T}|$ columns to $\mathcal{P}$ at each iteration.

The main issue is how to find negative reduced cost feasible patterns. Consider the dual variables associated to the constraints of the $R - SC_R$ model (see (3.11)-(3.17)). The reduced cost r_{k_t} of a given pattern variable λ_{k_t} is $c_{k_t} - \left[\boldsymbol{\mu^T}\ \boldsymbol{\nu^T}\right] \boldsymbol{a_{k_t}} - \left[\boldsymbol{\alpha^T}\ \boldsymbol{\beta^T}\ \epsilon\right] \overline{\mathbf{1}}_{\boldsymbol{t}}$, where $\boldsymbol{a_{k_t}} = [a^i_{k_t}]$ and $\overline{\mathbf{1}}_{\boldsymbol{t}}$ is a vector of size $2|\mathcal{T}| + 1$, with 1 in the rows corresponding to bin type t and in the last row, 0 otherwise. By (3.10), we expand this expression as follows

$$
\begin{aligned}
r_{k_t} &= C_t - \sum_{i \in \mathcal{I}^{\mathrm{NC}}} p_i \, a^i_{k_t} - \left[\boldsymbol{\mu^T} \ \boldsymbol{\nu^T} \right] \boldsymbol{a_{k_t}} - \left[\boldsymbol{\alpha^T} \ \boldsymbol{\beta^T} \ \epsilon \right] \boldsymbol{\bar{1}_t} \\
&= C_t - \sum_{i \in \mathcal{I}^{\mathrm{NC}}} p_i \, a^i_{k_t} - \sum_{i \in \mathcal{I}^{\mathrm{C}}} \mu_i \, a^i_{k_t} - \sum_{i \in \mathcal{I}^{\mathrm{NC}}} \nu_i \, a^i_{k_t} - \alpha_t - \beta_t - \epsilon \\
&= C_t - \sum_{i \in \mathcal{I}^{\mathrm{NC}}} (p_i + \nu_i) \, a^i_{k_t} - \sum_{i \in \mathcal{I}^{\mathrm{C}}} \mu_i \, a^i_{k_t} - \alpha_t - \beta_t - \epsilon. \qquad (3.28)
\end{aligned}
$$

We now define a column generation sub-problem. Given a bin of type $t \in \mathcal{T}$, this sub-problem finds the non-basic pattern with the minimum reduced cost. Note that, the vector $\boldsymbol{a_{k_t}}$ defining a not-yet-generated pattern $k_t \in \mathcal{K}_t$ is not known, but it may be expressed in terms of the item-to-bin assignment variable x_i, which is equal to 1 if item $i \in \mathcal{I}$ belongs to the pattern, 0 otherwise. Since the dual variables α_t, β_t, and ϵ, as well as the bin cost C_t, are constant for any given bin type $t \in \mathcal{T}$, then finding the feasible pattern with the minimum reduced cost for bin type $t \in \mathcal{T}$ becomes the following knapsack problem

$$
\begin{aligned}
\text{max} \quad & \left\{ \sum_{i \in \mathcal{I}^{\mathrm{NC}}} (p_i + \nu_i) \, x_i + \sum_{i \in \mathcal{I}^{\mathrm{C}}} \mu_i \, x_i \right\} && && (3.29) \\
\text{subject to} \quad & \sum_{i \in \mathcal{I}} w_i x_i \leq W_t && t \in \mathcal{T} && (3.30) \\
& x_i \in \{0, \ 1\} && i \in \mathcal{I}. && (3.31)
\end{aligned}
$$

Any feasible solution may be used to initialize the procedure, including the trivial solution obtained by loading each compulsory item into a different bin. More accurate heuristics are presented in Section 3.3.

Finally, note that a better lower bound can be obtained by taking the maximum between the two previous lower bounds. We call this new lower bound $LB_3 = \max\{LB_1, \, LB_2\}$.

3.3 Upper bounds

We derive several upper bounds to the GBPP: 1) through constructive heuristics, 2) by making feasible some of the GBPP lower bounds, and 3) through column

generation-based heuristics.

3.3.1 Upper bounds through constructive heuristics

We propose constructive heuristics to load items into bins based either on the FFD or the BFD heuristics for the BPP, with different sorting rules for items and bins. We briefly recall how FFD and BFD work. Starting with items sorted by non-increasing volume, FFD loads items one after the other into the first bin where they fit. BFD attempts to load each item into the "best" bin, usually the bin with the minimum *free volume* after loading the item. The free volume is defined as the bin volume minus the total volume of the loaded items. Both heuristics create a new bin when an item cannot be accommodated into the existing ones. Despite their simplicity, the FFD and BFD heuristics offer good performances for the BPP[Johnson et al., 1974, Martello and Toth, 1990].

Note that, whilst in the BPP items are sorted by non-increasing volume, many item and bin sorting rules are possible for the GBPP, due to the presence of multiple attributes. We exploit this characteristic in building our heuristics.

Given the sorted lists of items and bins, *SIL* and *SBL* (see Section 3.3.1 for deriving these lists), our heuristics are composed of three main components displayed in Algorithms 1, 2, and 3. Given a list S of selected bins (initially empty), for each item belonging to *SIL*, Algorithm 1 (the MAIN procedure) looks for the first or the best bin in S able to load such an item by using FFD or BFD, respectively. If this bin exists, the item is loaded into it, otherwise a new bin is possibly selected. Actually, a new bin is selected when the item is compulsory otherwise we evaluate whether to select a new bin or not. This evaluation is performed by Algorithm 2 (the PROFITABLE procedure), which measures the profitability of the current item. In particular, a new bin will be selected and the item will be loaded into it if the profit of this item plus the profits of the remaining non-compulsory items in SIL is greater than the cost of the new bin. Finally, the POST-OPTIMIZATION procedure of Algorithm 3 attempts to improve the final solution by evaluating possible bin swaps that replace loaded bins with available cheaper bins of sufficient capacity.

Algorithm 1 The MAIN procedure

$\mathcal{S} := \emptyset$
for all $i \in SIL$ **do**
  Identify the bin $b \in \mathcal{S}$ into which item i can be loaded:

- FFD: the first bin with enough empty volume to accommodate item i
- BFD: the bin with the minimum free volume after loading item i

  if b exists **then**
    Load item i into bin b
  else
    if $i \in I^C$ **then**
      Identify the first bin $b \in SBL \setminus \mathcal{S}$ such that $w_i \leq W_b$.
      Load item i into bin b
      $\mathcal{S} := \mathcal{S} \cup \{b\}$
    else
      Identify the bin $b \in SBL \setminus \mathcal{S}$ such that PROFITABLE(i, b) returns TRUE
      if b exists **then**
        Load item i into bin b
        $\mathcal{S} := \mathcal{S} \cup \{b\}$
      else
        reject item i
POST-OPTIMIZATION

Algorithm 2 The PROFITABLE procedure for new bin selection

SIL_i : sublist of SIL starting from the item i;
Load i into b and initialize the bin profit $P_b = p_i$;
for all $i' \in SIL_i$ **do**
  if i' can be loaded into b **then**
    Load i' into b and update the bin profit $P_b = P_b + p_{i'}$;
  if $P_b > c_b$, return TRUE **else** return FALSE.

Algorithm 3 The POST-OPTIMIZATION procedure

for all $j \in \mathcal{S}$ **do**
  for all $k \in \mathcal{J} \setminus \mathcal{S}$ **do**
    $U_j = \sum_{i \text{ loaded into } j} w_i$
    if $W_k \geq U_j$ and $C_k < C_j$ **then**
      Move all the items from j to k
      $\mathcal{S} = \mathcal{S} \setminus \{j\} \cup \{k\}$

Building the sorted lists of items and bins

We define four sorting rules to embed into the constructive heuristics we propose. All the rules have in common that compulsory items are sorted at the top of the item list by non-increasing volume. The four rules are:

1. ***Bins:*** Non-decreasing C_j/W_j and non-decreasing volumes W_j

   Non-compulsory items: Non-increasing p_i/w_i and non-increasing volumes w_i

2. ***Bins:*** Non-decreasing C_j/W_j and non-decreasing volumes W_j

   Non-compulsory items: Non-increasing volumes w_i and non-increasing p_i/w_i

3. ***Bins:*** Non-decreasing C_j/W_j and non-increasing volumes W_j

   Non-compulsory items: Non-increasing p_i/w_i and non-increasing volumes w_i

4. ***Bins:*** Non-decreasing C_j/W_j and non-increasing volumes W_j

   Non-compulsory items: Non-increasing volumes w_i and non-increasing p_i/w_i.

3.3.2 Upper bounds through the lower bound LB_1

To derive an upper bound from LB_1, introduced in Section 3.2.1, we apply our constructive heuristics to the two ordered lists of bins and items. We name this approach *lower bound-based constructive heuristics*. We refer to L-FFD and L-BFD when the constructive heuristics, which is based on the lower bound LB_1, implements the FFD and the BFD principle, respectively.

The ordered lists of items and bins are obtained as follows. An optimal solution of the AKP model (3.20)-(3.23) consists in a set of non-compulsory items, $\mathcal{I}_{agg}$, and a set of bins, $\mathcal{J}_{agg}$. $\mathcal{I}_{agg}$ contains the non-compulsory items associated to the variables x_i equal to 1 in the optimal solution of the AKP. Similarly, $\mathcal{J}_{agg}$ contains the bins associated to the variables y_j equal to 1 in the optimal solution of the AKP. We then extract two subsets $\mathcal{I}'_{agg}$ and $\mathcal{J}'_{agg}$ by selecting δ^i % of items in $\mathcal{I}_{agg}$ and δ^b% of bins in $\mathcal{J}_{agg}$, respectively. We then randomly select a particular sorting rule s among the four ones, and order the item and bin lists as follows

- Order the items in $\mathcal{I}'_{agg}$ according to s and put them at the top of the item list, before any non-compulsory item. Then, order the remaining items $\mathcal{I} \setminus \mathcal{I}'_{agg}$ according to s
- Order the bins in $\mathcal{J}'_{agg}$ according to s and put them at the top of the bin list. Then, order the remaining bins $\mathcal{J} \setminus \mathcal{J}'_{agg}$ according to s.

To state that L-FFD and L-BFD are characterized by δ^i, δ^b, and s, we write L-FFD$\left(\delta^i, \delta^b, s\right)$ and L-BFD$\left(\delta^i, \delta^b, s\right)$. The values of δ^i and δ^b are obtained by calibration (see Section 3.4.3 for details), whilst $s \in \{1, 2, 3, 4\}$, according to the four sorting strategies presented in Section 3.3.1.

3.3.3 Upper bounds through column generation-based heuristics

We present two approaches to compute upper bounds starting from the column generation-based solution of the relaxation *R-SC* of the set covering model *SC* (3.11)-(3.17).

The first approach is to solve *SC* exactly, e.g., by branch-&-bound, considering only the columns obtained by the column-generation procedure while computing the lower bound. This may still be quite time consuming, however. Consequently, we might stop with the branch-&-bound after a given computing time and name Z_{SC} the resulting value of the objective function, which is an upper bound to the GBPP.

The second approach is based on *diving*, a well-known method for finding good quality integer solutions from the optimal continuous solutions [Atamturk and Savelsberg, 2005]. The working principle is to iteratively round up variables and re-optimize the continuous relaxation.

The diving heuristics assumes that the optimal bin loading patterns of the GBPP are in the restricted set corresponding to the $R - SC_R$, and iteratively slightly perturbs the optimal continuous solution by fixing to integer some pattern-selection variables. The key feature is how to choose the variables to be fixed. Two strategies are obtained by selecting among the non-integral pattern variables the ones which maximize the expressions (3.32) and (3.33). This generates two diving heuristics named Diving1 and Diving2, which are included in the final comparative experiments

of Section 3.4:

$$\sum_{i \in \mathcal{I}^{\mathrm{NC}}} \nu_i \, a^i_{k_t} + \sum_{i \in \mathcal{I}^{\mathrm{C}}} \mu_i \, a^i_{k_t} \tag{3.32}$$

$$(1 - \lambda_{k_t}) \left(\sum_{i \in \mathcal{I}^{\mathrm{NC}}} \nu_i \, a^i_{k_t} + \sum_{i \in \mathcal{I}^{\mathrm{C}}} \mu_i \, a^i_{k_t} \right) \tag{3.33}$$

3.4 Computational results

The goal of the numerical experiments is to explore the performance of the proposed lower and upper bound procedures. Section 3.4.1 introduces the instance sets, whilst detailed results of different variants of the lower and upper bound procedures are given in Sections 3.4.2 and 3.4.3. We study the impact of a number of problem parameters on our best bounds in Section 3.4.4.

3.4.1 Instance classes

No instances are present in the literature for the GBPP. We generated instances, partially based on those for the VSBPP and the BPP [Monaci, 2002, Vanderbeck, 1996, Correia et al., 2008, Crainic et al., 2011]. The instances are grouped into 5 classes:

- Class 0: 300 instances by Monaci [Monaci, 2002]. We chose Monaci's instances because they are more challenging than Correia's [Correia et al., 2008], as shown in [Crainic et al., 2011]. Since these instances were conceived for the VSBPP, all items of each instance are compulsory. Ten instances were randomly generated for each combination of number of items, item profit, item volume, and bin type for a total of 300 instances. For the sake of completeness, we report here the details of Monaci's instances:
  - Number of items: 25, 50, 100, 200, and 500
  - Item volume: **I1:** [1, 100]; **I2:**[20, 100]; **I13:**[50, 100]
  - Item profit: not defined because all items are compulsory
  - Bin type:

  - 3 types of bins, with volumes 100, 120, and 150, respectively, and costs equal to volumes
  - 5 types of bins, with volumes 60, 80, 100, 120, and 150, respectively, and costs equal to volumes.

  For each bin type t, $L_t = 0$ and $U_t = \lceil V_{tot}/V_t \rceil$, where V_{tot} is the total item volume. No values for the total number of available bins U are given.

- Class 1: same instances of Class 0, but with all items non-compulsory and item profits generated according to the $p_i \in \lceil \mathcal{U}(0.5, 3)w_i \rceil$ uniform distribution.

- Class 2: same instances of Class 0, but with all items non-compulsory and item profit generated according to the $p_i \in \lceil \mathcal{U}(0.5, 4)w_i \rceil$ uniform distribution.

- Class 3: a selection of 12 large instances (500 items) from Class 1 and Class 2 with a representative mix of characteristics in terms of item volume, item profit, and bin types. For each instance, we randomly derived five more instances with 0%, 25%, 50%, 75%, and 100% of compulsory items, for a total of 60 instances.

- Class 4: the aim of this class is to study the behavior of Constraints (3.7) and (3.16) on the total number of available bins U. Thus, we select 24 instances from Class 1 and Class 2. For each instance, we first computed the number of bins $\overline{U}$ employed by the BFD constructive heuristics. We then solved the GBPP varying the value of U as a percentage of $\overline{U}$

$$U = \overline{U}(1 - X), \quad X \in \{0, 0.1, 0.2, 0.3\}, \tag{3.34}$$

  All these combinations make up Class 4 with 96 instances.

The algorithms were coded in C++ and the models implemented with CPLEX 12.1 ILOG Inc. [2009]. The upper bound Z_{SC} was computed using Gurobi 4.0, due to its efficiency in finding good feasible solutions within a quite limited computing time (20 seconds) Gurobi Optimization [2010]. Experiments were made on a Pentium IV 3.0 GHz workstation with 4 GB of RAM.

3.4.2 Lower bounds

Table 3.1 shows the lower bound results, comparing the performance of the three proposed lower bounds, LB_1, LB_2, and LB_3, relative to Z_{SC}, the best upper bound (Section 3.3.3) or to the known optimum solution of Class 0 instances, named in the following Monaci optima [Monaci, 2002]. Columns 1 to 3 display the instance class, number of bin types, and number of items, respectively. Columns 4 and 5, 6 and 7, and 8 and 9 display the mean percentage gap to Z_{SC} or the known optimum, and the number of optima achieved by LB_1, LB_2, and LB_3, respectively. Each row of Table 3.1 gives the results of 30 instances (3 item volume types, I1, I2, and I3, times 10 repetitions). For each class and globally, the table also displays the respective average gaps and the total number of optima attained (and the corresponding percentage with respect to the total number of instances).

Table 3.1 reports very promising results. The overall percentage gap for LB_3 is quite tight (0.08%) and almost half (46%) of the instances are solved to optimality.

3.4.3 Upper bounds

Table 3.2 displays comparative results for the constructive heuristics upper bounds. The first three columns display the same type of information as previously. Columns 4 to 7 and 8 to 11 display relative-gap results with respect to LB_3 (except for Class 0 instances with Monaci optima) for the FFD and the BFD procedures, respectively, using the four item and bin sorting rules of Section 3.3.1.

The results summed up in Table 3.2 show that BFD offers slightly better results than FFD. Furthermore, we see that BFD(3) is the best performing constructive heuristics.

We compare the remaining upper bounds, i.e., those obtained through the lower bound LB_1 (Section 3.3.2) and those derived from the column generation-based heuristics (Section 3.3.3) in Table 3.3. Column 1 shows the instance class, Column 2 the number of bin types, and Column 3 the number of items. For the remaining columns of Table 3.3, we have:

BFD(3): BFD heuristics with the third sorting rule

L-BFD(1, 0.1, 3): Upper bound obtained from the lower bound LB1 with the

			LB_1		LB_2		LB_3	
CLASS	**BINS**	**ITEMS**	**% GAP**	**OPT**	**% GAP**	**OPT**	**% GAP**	**OPT**
0	3	25	1.21	10	0.31	13	0.18	21
		50	0.65	12	0.21	5	0.13	16
		100	0.51	16	0.07	8	0.04	22
		200	0.31	19	0.04	5	0.02	23
		500	0.31	20	0.03	3	0.01	23
	5	25	0.80	10	0.20	13	0.12	20
		50	0.51	15	0.12	9	0.05	21
		100	0.49	18	0.07	6	0.02	22
		200	0.27	20	0.04	6	0.01	24
		500	0.25	20	0.01	6	0.00	24
			0.53	**160 (53%)**	**0.11**	**74 (25%)**	**0.06**	**216 (72%)**
1	3	25	2.16	4	0.27	16	0.19	20
		50	0.86	1	0.17	6	0.15	5
		100	0.72	2	0.12	3	0.11	5
		200	0.45	1	0.13	6	0.12	7
		500	0.33	0	0.10	3	0.10	3
	5	25	1.42	5	0.18	13	0.13	16
		50	0.75	3	0.10	12	0.09	13
		100	0.57	5	0.04	10	0.03	13
		200	0.29	4	0.03	6	0.02	9
		500	0.29	2	0.08	6	0.07	8
			0.78	**27 (9%)**	**0.12**	**81 (27%)**	**0.10**	**99 (33%)**
2	3	25	1.31	5	0.18	14	0.16	17
		50	0.61	4	0.12	4	0.10	8
		100	0.41	3	0.06	7	0.06	8
		200	0.30	1	0.10	5	0.10	5
		500	0.26	0	0.08	4	0.07	4
	5	25	0.98	4	0.12	14	0.09	16
		50	0.52	8	0.06	8	0.05	15
		100	0.40	6	0.04	6	0.03	11
		200	0.18	4	0.05	5	0.04	9
		500	0.19	1	0.05	5	0.05	6
			0.52	**36 (12%)**	**0.09**	**72 (24%)**	**0.07**	**99 (33%)**
OVERALL			**0.61**	**223 (25%)**	**0.11**	**227 (25%)**	**0.08**	**414 (46%)**

Table 3.1. Lower bound results

CLASS	BINS	ITEMS	FFD(1) % GAP	FFD(2) % GAP	FFD(3) % GAP	FFD(4) % GAP	BFD(1) % GAP	BFD(2) % GAP	BFD(3) % GAP	BFD(4) % GAP
0	3	25	12.80	12.80	3.68	3.68	12.80	12.80	3.47	3.47
		50	13.25	13.25	2.44	2.44	13.25	13.25	2.36	2.36
		100	12.21	12.21	1.66	1.66	12.21	12.21	1.62	1.62
		200	10.28	10.28	1.28	1.28	10.28	10.28	1.25	1.25
		500	8.97	8.97	1.08	1.08	8.97	8.97	1.07	1.07
	5	25	10.49	10.49	1.93	1.93	10.49	10.49	1.93	1.93
		50	11.59	11.59	1.79	1.79	11.59	11.59	1.78	1.78
		100	10.63	10.63	1.27	1.27	10.63	10.63	1.26	1.26
		200	10.82	10.82	0.83	0.83	10.82	10.82	0.82	0.82
		500	10.44	10.44	0.65	0.65	10.44	10.44	0.63	0.63
			11.15	**11.15**	**1.66**	**1.66**	**11.15**	**11.15**	**1.62**	**1.62**
1	3	25	13.18	17.03	4.07	8.89	12.89	16.67	3.96	8.72
		50	13.62	17.93	3.22	7.59	13.57	17.87	3.20	7.54
		100	12.67	16.60	2.18	7.36	12.55	16.52	2.16	7.34
		200	11.30	15.08	1.47	7.15	11.26	15.05	1.45	7.13
		500	9.83	13.44	0.94	6.58	9.81	13.46	0.94	6.58
	5	25	9.74	14.86	3.38	8.40	9.79	14.54	3.33	8.37
		50	10.56	16.12	3.49	7.60	10.56	15.88	3.46	7.56
		100	9.53	14.41	1.99	6.41	9.49	14.32	1.97	6.39
		200	9.57	14.84	1.48	6.28	9.55	14.77	1.49	6.27
		500	9.36	14.68	1.00	6.32	9.36	14.65	1.00	6.32
			10.93	**15.50**	**2.32**	**7.26**	**10.88**	**15.38**	**2.30**	**7.22**
2	3	25	9.50	10.25	3.45	4.63	9.48	10.22	3.17	4.49
		50	10.91	11.99	2.39	4.58	10.85	11.93	2.32	4.54
		100	9.42	10.83	1.43	3.54	9.35	10.79	1.41	3.53
		200	8.35	9.57	1.20	3.37	8.31	9.58	1.20	3.33
		500	7.37	8.81	0.77	3.30	7.33	8.82	0.77	3.29
	5	25	7.18	9.36	3.28	3.86	7.18	9.26	3.35	3.86
		50	7.45	9.65	2.47	3.33	7.47	9.57	2.31	3.32
		100	6.77	9.11	1.73	3.59	6.75	9.08	1.70	3.58
		200	6.78	9.30	1.13	3.26	6.77	9.25	1.12	3.25
		500	6.56	9.16	0.77	3.13	6.56	9.16	0.77	3.13
			8.03	**9.80**	**1.86**	**3.66**	**8.00**	**9.77**	**1.81**	**3.63**
OVERALL			**10.04**	**12.15**	**1.95**	**4.19**	**10.01**	**12.10**	**1.91**	**4.16**

Table 3.2. Constructive heuristics upper bounds

constructive heuristics BFD(3) and $\delta^i = 1$, $\delta^b = 0.1$. The values of δ^i and δ^b were obtained by calibration performed by running L-BFD$\left(\delta^i, \delta^b, 3\right)$ on a number of selected instances, and varying δ^i and δ^b from 0.1 to 1 with a step of 0.1. The pair $\left(\delta^i, \delta^b\right)$ which gave the minimum mean gap was then selected

C-BFD(3)= min $\left\{\text{BFD(3)}, \min_{\delta^i \in \Delta^i, \delta^b \in \Delta^b} \left\{\text{L-BFD}\left(\delta^i, \delta^b, 3\right)\right\}\right\}$, where $\Delta^i = \Delta^b = \{0.1, 0.2, 0.3\}$; These value combinations provided low mean gaps during the calibration phase

Z_{SC} : Upper bound obtained through the column generation-based heuristics

DIVE(1), DIVE(2): Upper bounds obtained through the column generation-based heuristics using the diving strategies Diving1 and Diving2, respectively (Section 3.3.3)

B-DIVE(1)= Minimum {BFD(3), DIVE(1) }

B-DIVE(2)= Minimum {BFD(3), DIVE(2) }.

As far as computing times are considered, detailed results, available from the authors, show that they are generally insignificant for small-size instances. For larger instances (500 items), BFD(3), L-BFD(1, 0.1, 3), and C-BFD(3) require computing times of less than 0.1 seconds. DIVE(1) and DIVE(2) require computing times of 1 second, while B-DIVE(1) and B-DIVE(2) take about 0.3 seconds. Z_{SC} is the most time consuming heuristics with a computing time that may go to the time limit of 20 seconds. We further discuss computing-time issues for Z_{SC} in Section 3.4.4.

Fast solutions can thus be obtained through the BFD(3) heuristics, but the quality is not very good since the overall average gap is 1.91%. L-BFD(1, 0.1, 3) is, in principle, worse than BFD(3), with a gap of 2.18%. But, if exploited to compute the C-BFD(3) heuristics, the quality improves and the gap reduces to 1.58%. The best results are yielded by Z_{SC} with an overall gap of 0.11%. Nevertheless, as mentioned before, this is also the most time consuming heuristic. A good compromise between accuracy and efficiency is offered by the diving strategies (DIVE(1), DIVE(2), B-DIVE(1), and B-DIVE(2), with average gaps varying between 0.53% and 1.22%.

CLASS	BINS	ITEMS	BFD(3) % GAP	L-BFD(1, 0.1, 3) % GAP	C-BFD(3) % GAP	Z_{SC} % GAP	DIVE(1) % GAP	DIVE(2) % GAP	B-DIVE(1) % GAP	B-DIVE(2) % GAP
		25	3.47	3.35	1.94	0.26	1.95	2.02	1.31	1.54
		50	2.36	2.55	1.84	0.19	1.29	1.39	0.95	1.05
	3	100	1.62	1.53	1.15	0.16	1.85	1.45	0.84	0.66
		200	1.25	1.34	1.02	0.18	1.75	1.15	0.59	0.51
0		500	1.07	0.85	0.68	0.31	2.12	1.04	0.59	0.51
		25	1.93	2.49	1.75	0.13	1.27	0.88	0.68	0.60
		50	1.78	2.56	1.68	0.06	0.79	0.55	0.63	0.52
	5	100	1.26	1.84	1.17	0.03	0.59	0.45	0.42	0.39
		200	0.82	1.57	0.82	0.06	0.64	0.45	0.34	0.28
		500	0.63	1.25	0.63	0.05	0.76	0.41	0.16	0.14
			1.62	**1.93**	**1.27**	**0.14**	**1.30**	**0.98**	**0.65**	**0.62**
		25	3.96	3.85	2.95	0.19	1.30	1.42	0.87	0.89
		50	3.20	2.98	2.46	0.15	1.05	0.94	0.88	0.82
	3	100	2.16	2.37	1.98	0.11	0.81	2.18	0.66	0.68
		200	1.45	1.77	1.37	0.12	1.57	2.28	0.55	0.61
1		500	0.94	1.14	0.92	0.10	1.89	2.79	0.40	0.55
		25	3.33	3.67	2.42	0.12	0.74	0.76	0.55	0.58
		50	3.46	3.85	2.92	0.09	0.37	0.52	0.37	0.52
	5	100	1.97	2.47	1.83	0.03	0.53	0.52	0.35	0.36
		200	1.49	1.92	1.45	0.02	0.55	1.11	0.28	0.32
		500	1.00	1.21	0.97	0.07	0.73	1.18	0.15	0.18
			2.30	**2.52**	**1.93**	**0.10**	**0.95**	**1.37**	**0.51**	**0.55**
		25	3.17	3.10	2.09	0.16	1.99	2.07	1.05	0.95
		50	2.32	2.70	2.09	0.10	0.83	1.25	0.77	0.84
	3	100	1.41	1.67	1.32	0.06	0.98	1.64	0.38	0.48
		200	1.20	1.49	1.15	0.10	1.37	2.12	0.41	0.52
2		500	0.77	0.94	0.75	0.07	1.36	2.50	0.24	0.40
		25	3.35	3.52	2.71	0.09	0.48	0.66	0.43	0.61
		50	2.31	2.86	2.07	0.05	0.47	0.46	0.37	0.31
	5	100	1.70	1.96	1.53	0.03	0.21	0.43	0.21	0.30
		200	1.12	1.50	1.07	0.04	0.36	0.87	0.25	0.28
		500	0.77	1.01	0.76	0.05	0.46	1.20	0.13	0.17
			1.81	**2.08**	**1.56**	**0.07**	**0.85**	**1.32**	**0.42**	**0.49**
OVERALL			**1.91**	**2.18**	**1.58**	**0.11**	**1.04**	**1.22**	**0.53**	**0.55**

Table 3.3. Upper bound comparisons

3.4.4 Sensitivity analysis

This subsection is dedicated to the analysis of the impact on the performance of best bound procedures on a number of important problem parameters: the percentage of compulsory items, the total number of available bins. and the variability of item volumes and profits.

Percentage of compulsory items

We consider the instances of Class 3, where 12 large instances (500 items) taken from Class 1 and Class 2 are selected for each percentage of compulsory items set at 0%, 25%, 50%, 75%, and 100%. Table 3.4 display comparative results for three GBPP upper bounds:

BEST BFD: Best among the constructive heuristics BFD(3), L-BFD(1, 0.1, 3), and C-BFD(3)

$\mathbf{Z_{SC}}$: Our best upper bound

BEST DIVING: Best between the diving heuristics B-DIVE(1) and B-DIVE(2).

Table 3.4a displays the mean percentage gap between these upper bounds and the lower bound LB_1 over the 12 instances, while Table 3.4b lists the corresponding computing times in seconds.

The results clearly show a trend. The most difficult instances for all the upper bounds are those where compulsory and non-compulsory items are more or less of equal quantity. All upper bounds perform better when one type of items (compulsory or non-compulsory) dominates the other. The ranking among the different upper bounds previously observed is confirmed by these results: the best constructive heuristics is extremely fast but yields worse results than the best diving method, which is outperformed by the Z_{SC}.

Total number of available bins

To test the impact of the total number of available bins on the accuracy of the proposed bounds, we computed the relative gaps between our best upper bound Z_{SC} and the lower bounds LB_1, LB_2, and LB_3. Computations were performed on

% COMPULSORY ITEMS	BEST BFD % GAP	Z_{SC} % GAP	BEST DIVING % GAP
0	0.77	0.12	0.37
25	1.73	0.51	1.00
50	12.21	2.72	7.52
75	2.04	0.52	1.15
100	0.85	0.16	0.51
MEAN	**3.52**	**0.81**	**2.11**

(a)

% COMPULSORY ITEMS	BEST BFD (seconds)	Z_{SC} (seconds)	BEST DIVING (seconds)
0	< 0.01	11.37	0.39
25	< 0.01	11.94	0.55
50	< 0.01	13.95	0.42
75	< 0.01	13.41	0.33
100	< 0.01	13.42	0.25
MEAN	$<$ **0.01**	**12.82**	**0.39**

(b)

Table 3.4. Impact of the percentage of compulsory items on the best upper bounds

the 24 instances of Class 4. Recall that these instances were selected from Classes 1 and 2 for which we reduced the total number of available bins U by a percentage ranging from 0% up to 30%, with a step of 10%.

Figure 3.1 displays the gaps after 20 seconds of computing time. The behaviour illustrated by the figure is that the tighter the constraint on the total number of available bins, the more the accuracy of the gap degrades. The question then is whether this degradation follows from inaccuracies in the values of the upper bounds or from the tightness of the constraints on the bin supply. To start answering this question, we need Z_{SC} to be as close as possible to the optimal solution for most instances; 20 seconds is too short for most cases, however. We therefore let computations continue until a time limit of 1000 seconds. The corresponding gaps between Z_{SC} and the three lower bounds are displayed in Figure 3.2.

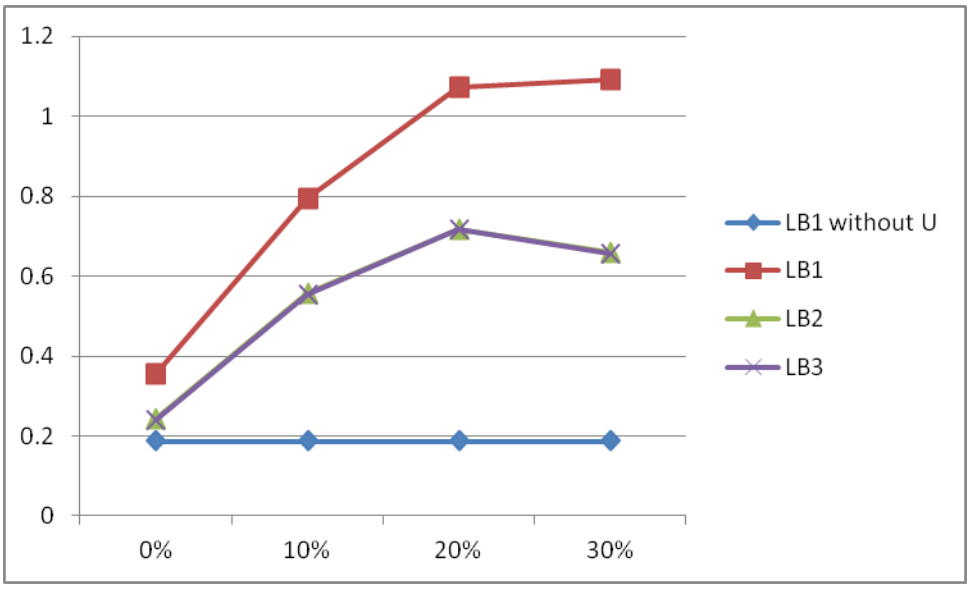


Figure 3.1. Class 4 gaps between Z_{SC} and LB_1, LB_2, and LB_3 for varying U with a time limit of 20 seconds

The trend is similar to that of Figure 3.1 with just a small reduction (0.3%) in the gap values. As the values of the upper bound Z_{SC} are now close or equal to the optimal ones, we conclude that the gap degradation is not due to the accuracy of Z_{SC}, but to the impact of the constrains on the total number of available bins.

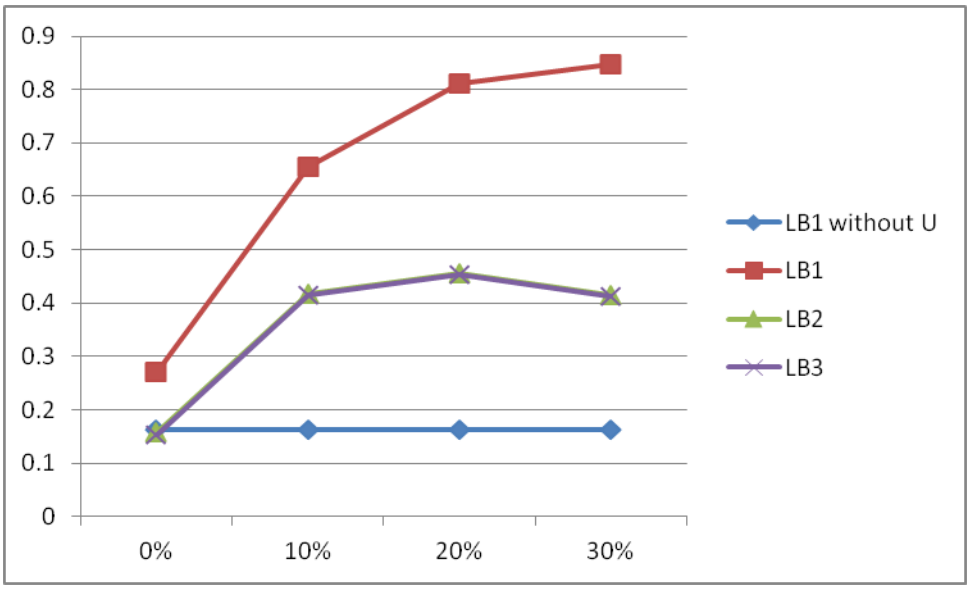


Figure 3.2. Class 4 gaps between Z_{SC} and LB_1, LB_2, and LB_3 for varying U with a time limit of 1000 seconds

Impact of item volumes and profits

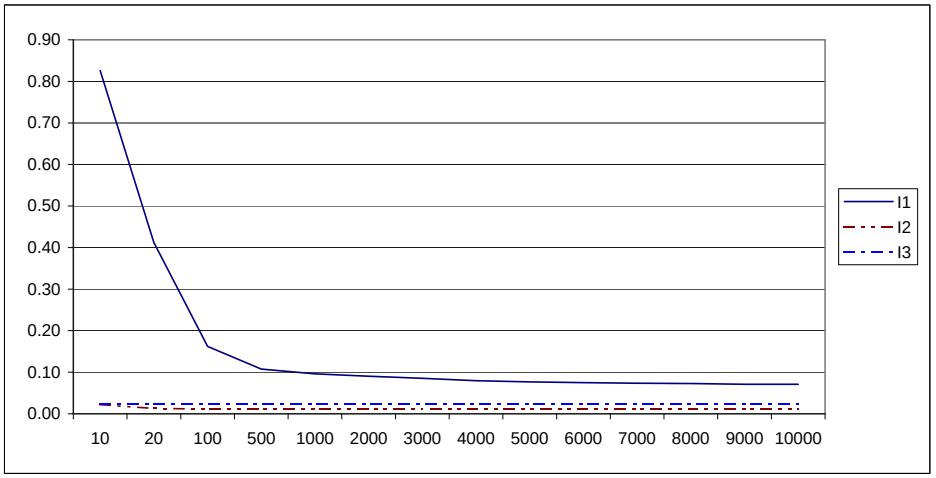
The last set of experiments had a double objective. First, to inquire whether longer computation times may improve the accuracy of Z_{SC}. Second, to evaluate the impact of the variability in item volumes and profits on the performance of the same upper bound procedure.

Experiments were thus performed by extending the time limit of the Z_{SC} procedure to 10000 seconds. All instances with up to 200 items were solved to optimality in at most 60 seconds, independently of the parameters used to generate the instance data. The behavior changed when the number of items was increased to 500, the time required to reach the best solutions and the rate of improvement varying with the problem characteristics.

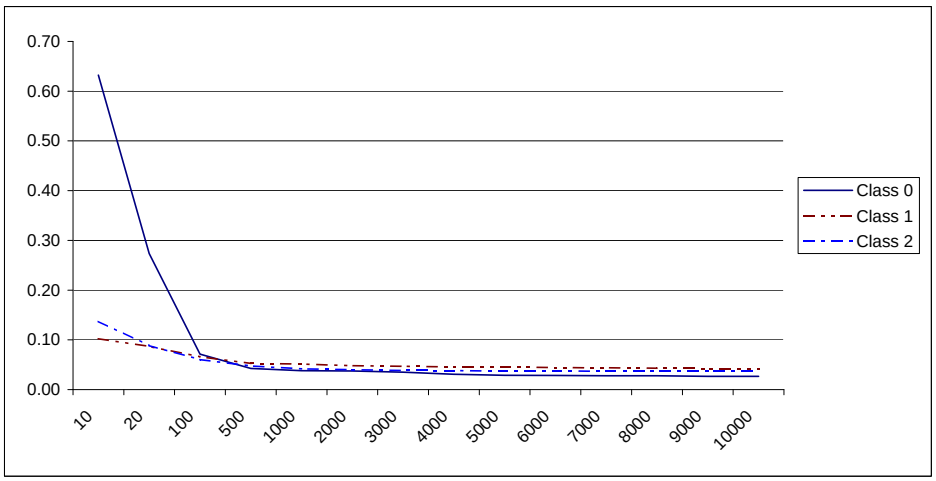
Figures 3.3a and 3.3b plot the evolution of the mean percentage gap between Z_{SC} and LB_1, when the computing time is increased. Instances are grouped by item volume (I1, I2, and I3) in Figure 3.3a, whilst the item profit taken from Classes 0, 1, and 2 is used to group instances in Figure 3.3b.

The results indicate that the most challenging instances are characterized by a large variance in item volume (I1), the method achieving very rapidly extremely good results on all other problem instances. In this case, see Figure 3.3a, the Z_{sc} heuristics achieves rapidly (around 100 seconds) a gap of less than 0.2%, requires four times to drop to 0.1%, and then, until the 10000 seconds time limit, continues to drop slowly to a gap of less than 0.1%. After this time limit, the convergence process slows down considerably. This behavior is partially explained by the fact that the column generation generates a large number of patterns when there is significant variation in item volumes. Then, the general-purpose branch & bound software used has to consider a large number of variables, causing the reduction of its convergence rate. Moreover, a wide choice in terms of patterns also degrades the lower bound precision.

Not surprisingly, see Figure 3.3b, Z_{SC} is most challenged by problem instances not displaying the characteristics the bounding procedures were developed for. In our case, these instances are in Class 0 where all items are compulsory and for which the procedure requires about 300 seconds on average to reach a gap of less than 0.1%. It is very encouraging, however, to observe that, not only the procedure



(a)



(b)

Figure 3.3. Mean Z_{SC} - LB_1 gap versus computing time for 500-item instances

performs very well on GBPP problem instances, but its behavior is also good on instances lacking the characteristics it was designed for.

Chapter 4

Branch-and-price and Beam Search for the Generalized Bin Packing Problem

4.1 Introduction

In this chapter, we give an exact method, based on branch-and-price, for solving the GBPP. Our method is characterized by a two-layer branching strategy – first on the bins and then on the items – instead of a simple item to bin assignment as previously done in the bin packing literature [Martello and Toth, 1990, Monaci, 2002]. This exact technique allows us to reach a mean gap of 0.03% and close most of the instances in the GBPP literature. Exploiting the branch-and-price skeleton, we then propose an approximate method, named beam search, which visits a portion of the branch-and-price tree only.

Extensive computational tests obtained by varying the beam search parameters allow us to find results comparable to the branch-and-price within a limited computing time.

This chapter is organized as follows. In Section 4.2 we thoroughly discuss the branch-and-price algorithm and in Section 4.3 the beam search heuristics. These algorithms are both extensively tested in Section 4.4.

4.2 Branch-and-price

The branch-and-price [Barnhart et al., 1998] is an exact method which aims to find an optimal solution of an integer linear problem by exploiting a tree structure where an easier sub-problem is solved at each node. It is a development of the branch-and-bound method [Lawler and Wood, 1966] with the addition of a column generation procedure at each node. Our branch-and-price is based on the SC model (3.11)-(3.17). At each node, we solve a sub-problem consisting in its continuous relaxation R-SC with the addition of proper node constraints. Each sub-problem is solved through a column generation procedure, as illustrated in Section 3.2.2.

In the following, we name $LB(u)$ and $UB(u)$ respectively the lower and the upper bound associated to the sub-problem of node u (also called Master Problem of node u), and UB the global upper bound to the problem. Note that $LB(0) = LB_3$ since, at the root node of the search tree (node 0), the best lower bound is LB_3. We developed our branch-and-price algorithm for the GBPP extending the ideas of Bettinelli et al. [2010] who proposed a branch-and-price technique for the VCSBPP with minimum filling constraints.

4.2.1 Bounds at the root node

At the root node we compute the lower bounds LB_1, LB_2, LB_3, and the upper bounds BFD and Z_{SC}, as described in Section 3.3.

4.2.2 Branching

We adapted to the GBPP the branching strategy of Bettinelli et al. [2010]. At each branching node we perform a binary branching through two criteria which consider the patterns created by the column generation at that node. The first criterion involves the number of bins for each bin type $t \in \mathcal{T}$. If it cannot be adopted (see below) then we move to the second criterion, which works on the items. In Monaci [2002], the author proposed another kind of branching based on the assignment of critical items into bins, but, after preliminary experiments, this approach turned out not to be very effective.

Branching on the number of bins

Given the patterns created by the column generation when solving the *R-SC* model, we compute $z_t = \sum_{k \in \mathcal{K}_t} \lambda_k$ and we consider the bin type t^* such that z_{t^*} has its fractional part the closest to 0.5. Then, in the first child node, we impose the additional constraint to use at least $L_{t^*} = \lceil z_{t^*} \rceil$ bins of type t^*, whilst in the second child node we impose the additional constraint to use at most $U_{t^*} = \lfloor z_{t^*} \rfloor$ bins of type t^*. If t^* does not exist we consider the second criterion, which branches on the items.

Branching on the items

Given the patterns created by the column generation when solving the *R-SC* model, we compute $f_{ij} = \sum_{t \in \mathcal{T}} \sum_{k \in \mathcal{K}_t : a_k^i = 1 \wedge a_k^j = 1} \lambda_k$ and we select the items i^* and j^* such that $f_{i^* j^*}$ is the closest to 0.5. The additional branching constraints are then

$$x_{i^*} = x_{j^*} \tag{4.1}$$

in the first child node and

$$x_{i^*} + x_{j^*} \leq 1 \tag{4.2}$$

in the second child node. Let us note that constraints (4.1) and (4.2) are not explicitly added to each node. As we show in Section 4.2.3, they are implicitly managed within the oracle in the pricing step.

Let us observe that (4.1) means that items i^* and j^* must be loaded together in the same bin, otherwise they are not loaded at all. Vice versa, (4.2) states that items i^* and j^* cannot appear together in the same bin. Note that the presence of constraints (4.2) changes the type of pricing sub-problem, having to face a Knapsack Problem with Conflict Graph (also named Disjunctively Constrained Knapsack Problem), a variant of the standard Knapsack Problem much difficult to solve [Hifi and Michrafy, 2007]. Conversely, constraints (4.1) can be implicitly satisfied substituting the involved items by a macro item, say h, which volume w_h is the total volume of the items, profit p_h is the total profit of the non-compulsory items, and which dual variable π_h is the total of the dual variables of the items. This macro

item becomes compulsory if at least one of its items is compulsory.

4.2.3 Pricing

Pricing at a given node, say u, is performed by applying a column generation technique to try to tighten the lower bound of node u, $LB(u)$.

As stated in Section 4.2.2, the pricing sub-problem at non-root nodes can be a Knapsack Problem with Conflict Graph. Due to the high computational time required to optimally solve this problem, three oracles with increasing computational time are used. The first and the second oracles are simpler and faster than the third one, but they can fail. The third oracle never fails but it is the most time consuming one. If the first or the second oracle succeeds, we quit the sub-problem, otherwise we go to the next oracle. This particular architecture of the sub-problem limits the third oracle usage in order to reduce the computing time. In particular, the three oracles are:

- Heuristic oracle
- Knapsack Problem without constraints (4.2)
- Knapsack Problem with constraints (4.2).

We remind that constraints (4.1) are implicitly managed in the three oracles through the introduction of macro items (see Section 4.2.2), therefore, only constraints (4.2) may appear when solving the oracles. The first sub-problem, the heuristic oracle, is a greedy procedure which produces a pattern by first sorting items by non-increasing values of $\frac{\pi_h}{w_h}$ and then by trying to insert the sorted items into a bin of the current type $t \in \mathcal{T}$. Note that this oracle may fail due to two reasons: a) the loaded items violate one of the additional constraints (4.2) (which means that the new pattern is infeasible) or b) the oracle generated a pattern with a positive reduced cost. Failure b) is a drawback due to the heuristic nature of the oracle. Indeed, since this oracle is not exact, it does not generate, in principle, a pattern yielding the *minimum* reduced cost. Therefore, if the first oracle generates a negative reduced cost pattern, we however have (although it is not the one yielding the minimum reduced cost) a profitable pattern for proceeding with the column

generation procedure and so we can quit the sub-problem. Vice versa, if the first oracle generates a positive reduced cost pattern then, since it is not the pattern yielding the minimum reduced cost, there could exist, however, a negative reduced cost pattern. Since, in this particular case, we cannot predict whether such a negative reduced cost pattern exists, the first oracle fails and we move to the second one.

The second oracle consists in solving a Knapsack Problem on the items. without constraints (4.2). Since this is an exact oracle, it fails only if constraints (4.2) are violated. Hence, if the solution satisfies these constraints we are done. Otherwise two things may happen: a) the solution is not feasible but its reduced cost is positive, b) even the second oracle fails if at least one among constraints (4.2) is violated. In the first case, since this is an exact sub-problem, it means that also the remaining patterns have positive reduced costs, even if the created pattern is infeasible. Therefore we quit. In the second case, we undergo oracle three.

The third oracle consists in solving a Knapsack Problem with constraints (4.2). By construction, it never fails. Nevertheless, the presence of constraints (4.2) makes it time consuming. That is why we leave this oracle at the end, after the first two oracles. Computational experience confirms that the third oracle is actually rarely used.

To speed-up the whole pricing procedure, we exploit the fact that the lower bound of a child node cannot be less than the lower bound of its father node. In other words, let $u-1$ be the father node of node u (different from the root node), then $LB(u) \geq LB(u-1)$. This implies the addition to the Master Problem of node u the following constraint:

$$\sum_{t \in \mathcal{T}} \sum_{k \in \mathcal{K}_t} c_k \lambda_k \geq LB(u-1). \tag{4.3}$$

Note that the introduction of (4.3) in the Master Problem of node u modifies the oracle (3.29)-(3.31). Let $\theta \geq 0$ be the dual variable associated to constraint (4.3) then, following the same procedure presented in Section 3.2.2, the new column-generation sub-problem becomes:

$$
\begin{aligned}
\max \quad & \left\{ \sum_{i \in \mathcal{I}^{\mathrm{NC}}} [(1-\theta)p_i + \nu_i]\, x_i + \sum_{i \in \mathcal{I}^{\mathrm{C}}} \mu_i\, x_i \right\} & \\
\text{subject to} \quad & \sum_{i \in \mathcal{I}} w_i x_i \leq W_t & t \in \mathcal{T} \\
& x_i \in \{0,\ 1\} & i \in \mathcal{I}
\end{aligned}
$$

4.2.4 Rounding

This technique tries to tighten the lower bound yielded by the pricing procedure. Let $LB_2(u)$ be the lower bound produced by the column generation at node u, then a new lower bound can be found solving the following problem:

$$
\begin{aligned}
\min \quad & LB(u) = \sum_{j \in \mathcal{J}} C_j y_j - \sum_{i \in \mathcal{I}^{\mathrm{NC}}} p_i x_i & & (4.4) \\
\text{subject to} \quad & \sum_{j \in \mathcal{J}} C_j y_j - \sum_{i \in \mathcal{I}^{\mathrm{NC}}} p_i x_i \geq \lceil LB_2(u) \rceil & & (4.5) \\
& \sum_{i \in \mathcal{I}^{C}} w_i + \sum_{i \in \mathcal{I}^{NC}} w_i x_i \leq \sum_{j \in \mathcal{J}} W_j y_j & & (4.6) \\
& L_t \leq \sum_{j \in \mathcal{J}:\sigma(j)=t} y_j \leq U_t & t \in \mathcal{T} & (4.7) \\
& \sum_{j \in \mathcal{J}} y_j \leq U & & (4.8) \\
& x_i \in \{0,\ 1\} & i \in \mathcal{I}^{\mathrm{NC}} & (4.9) \\
& y_j \in \{0,\ 1\} & j \in \mathcal{J}, & (4.10)
\end{aligned}
$$

where L_t and U_t are the bounds on the number of bins which have been previously calculated in the branching step. This problem is based on the AKP presented in Section 3.2.1, as it can be seen from (4.4) and (4.6), which, respectively, play the same role of (3.19) and (3.20). Here, the main idea is to try to increase the lower bound $LB_2(u)$ yielded by the pricing step. This is expressed by constraint (4.5). Finally, through this problem, we also try to tighten the global upper bound by solving a BFD heuristics with exactly $\sum_{j \in \mathcal{J}:\sigma(j)=t} y_j$ bins for each bin type $t \in \mathcal{T}$ and

considering the disjoint additional constraints on the items.

4.3 Beam search

Beam search is a particular heuristics that relies on a branch-and-bound or branch-and-price tree [Della Croce et al., 2004]. The approximation behavior is due to the fact that just a part of the search tree is explored. This means that, at a given level of the tree, only γ nodes are visited. The parameter γ is the size of the beam. The γ nodes are selected according to a particular criterion. In our tests we have considered a beam size up to 4 and selected those nodes showing the best absolute gaps, computed as $|LB(u) - UB(u)|$. Since the philosophy we adopted when developing the beam search was to save time, we decided to skip the Z_{SC} computation and the rounding problem at each node.

4.4 Computational results

In this section, we present the computational results of our branch-and-price and beam search methods. First, the testing environment is presented in Subsection 4.4.1, while detailed computational results of the branch-and-price and the beam search are given in Subsection 4.4.2. Finally, being the GBPP a generalization of the VCSBPP, in Subsection 4.4.3 we compare the results of the branch-and-price and the beam search with the state-of-the-art methods for the VCSBPP in order to show how much the generalization process affects the results both in terms of efficiency and accuracy.

4.4.1 Testing environment

The algorithms were coded in C++ and the models implemented with CPLEX 12.1 [ILOG Inc., 2009]. Z_{SC} was again computed within a limited computing time of 20 seconds, when needed. We ran our branch-and-price algorithm with a time limit of one hour and our beam search with a time limit of three minutes. Experiments were conducted on a Pentium IV 3.0 GHz workstation with 4 GB of RAM.

4.4.2 GBPP results

In Table 4.1, we report the branch-and-price results for classes 0, 1, and 2. In particular, column 1 shows the class number; column 2 the number of bin types; column 3 the number of items, column 4 the percentage gap at the root node, column 5 the residual percentage gap at the end of the branch-and-price; column 6 the number of visited nodes on average, column 7 the number of instances solved to optimality over 900; column 8 the number of instances solved to optimality where the solution found at the root node is also an optimal solution; column 9 the average computing time. Note that the percentage gap at the root node is computed as the difference between the best lower and upper bound at the root node over the best lower bound at the root node; i.e. $\left|\frac{UB(0)-LB(0)}{LB(0)}\right| \cdot 100$. Note that, since $LB(0)$ can be negative, we compute the gap with absolute values. If $LB(0) = 0$, the gap is set equal to $UB(0)$.

To compute the residual gap at the end of the branch-and-price, we define the best lower bound at the end of the branch-and-price LB_B as follows:

$$LB_B = \begin{cases} UB & \text{if the best solution found so far is optimal} \\ LB(0) & \text{otherwise.} \end{cases}$$

Then the residual percentage gap is computed as $\left|\frac{UB-LB_B}{LB_B}\right| \cdot 100$, where UB is the upper bound corresponding to the best solution found by the branch-and-price.

The results of Table 4.1 are quite satisfactory: not only we reduce the gap from 0.13% (i.e. the gap calculated at the root node) to 0.03%, but we also solve to optimality 702 instances over 900. The most difficult instances to solve are those with 500 items, and in particular those with 3 bin types. This is justified by the fact that the more the number of items increases, the more the instances are difficult to solve. Moreover, with 3 bin types the choice on the available bins is quite reduced. This makes the problem harder due to the presence of equivalent patterns which increase both the number of variables involved in any column generation iteration and the fragmentation of these variables in the optimal solution of the pricing procedure.

In Table 4.2, the branch-and-price results for Class 3 are presented. We decided

to separate Class 3 results from the other classes because these instances are characterized by the presence of both compulsory and non-compulsory items, while the number of items is always 500. Therefore there is not a direct matching with the columns of Table 4.1. In Table 4.2, the columns have the following meaning: column 1 shows the percentage of compulsory items; column 2 the percentage gap at the root node; column 3 the residual percentage gap after the branch-and-price; column 4 the number of visited nodes on average; column 5 the number of instances solved to optimality over 60; column 6 the number of instances solved to optimality where the solution found at the root node is also an optimal solution; column 7 the average computing time.

The percentage gap at the root node and the residual gap at the end of the branch-and-price are computed as for Table 4.1. In this case, we solved to optimality 19 instances over 60, i.e. 31% of Class 3 instances. Although the absolute difference of the gap reduction is approximately the same in the two tables (around 0.1%), the residual gap is not as good as in Table 4.1. This is justified by two issues. First one, the gap at the root node is already high. This is justified by the fact that, for large size instances, 20 seconds of time limit are not enough to compute Z_{SC} to optimality. This implies a higher bound at the root node. The second issue concerns the fact that, as in Class 3 instances, both compulsory and non-compulsory items are present, two different sets of constraints are necessary: (3.12) for compulsory items and (3.13) for non-compulsory items. This splitting of items with their relative constraints makes the problem harder to solve and justifies the gap growth for Class 3 instances.

In Table 4.3, we report our beam search results. In particular, the columns have the following meaning: column 1 shows the class number; column 2 the beam size; column 3 the residual percentage gap after applying the beam search; column 4 the number of instances solved to optimality over 960; column 5 the number of solutions better than those found by the branch-and-price and, finally, column 6 the average computing time. In this table, we report all the classes together because we aim to show the overall gap depending on the beam size rather than on the instance attributes. The residual percentage gap is computed in a similar way as for the branch-and-price. Indeed, due to the previous branch-and-price calculation, now we know the optima of many instances and we can refer to them when computing

CLASS	TYPES	ITEMS	% GAP(0)	% GAP	NODES	OPT	ROOT OPT	TIME
		25	0.27	0.00	5.00	30	22	0.05
		50	0.21	0.00	26.33	30	19	0.51
	3	100	0.24	0.02	1190.93	28	13	80.62
		200	0.18	0.07	4107.80	19	9	1057.24
0		500	0.25	0.20	901.67	13	7	2165.01
		25	0.14	0.00	9.93	30	25	0.09
		50	0.10	0.00	13.07	30	22	0.31
	5	100	0.13	0.01	776.53	29	11	146.84
		200	0.09	0.05	2970.27	22	13	680.71
		500	0.06	0.03	1008.80	16	9	1908.28
			0.17	**0.04**	**1101.03**	**247**	**150**	**603.97**
		25	0.32	0.00	13.80	30	20	0.20
		50	0.16	0.00	188.67	30	13	22.41
	3	100	0.13	0.04	3297.87	19	6	963.22
		200	0.09	0.03	3607.33	21	5	1115.82
1		500	0.21	0.21	1099.80	10	5	2560.55
		25	0.20	0.00	100.07	30	23	9.16
		50	0.06	0.00	429.73	30	24	45.65
	5	100	0.05	0.01	1939.00	24	12	625.94
		200	0.03	0.01	4322.93	18	6	1199.36
		500	0.03	0.03	933.47	14	9	2053.72
			0.13	**0.03**	**1593.27**	**226**	**123**	**859.60**
		25	0.15	0.00	13.20	30	22	0.30
		50	0.19	0.01	797.27	28	17	222.94
	3	100	0.07	0.01	2246.07	22	9	744.96
		200	0.07	0.04	4593.00	19	7	1209.31
2		500	0.21	0.19	1030.80	11	6	2404.29
		25	0.07	0.00	23.07	30	26	1.81
		50	0.06	0.01	726.67	28	19	106.84
	5	100	0.03	0.01	1974.00	23	13	861.03
		200	0.02	0.01	3462.60	22	6	1084.04
		500	0.02	0.02	836.53	16	11	1959.58
			0.09	**0.03**	**1570.32**	**229**	**136**	**859.51**
OVERALL			**0.13**	**0.03**	**1421.54**	**702**	**409**	**774.36**

Table 4.1. Branch-and-price results for Classes 0, 1, and 2

PERC.	% GAP(0)	% GAP	NODES	OPT	ROOT OPT	TIME
0	0.11	0.10	1291.33	3	1	2820.44
25	0.32	0.31	1109.00	4	3	2472.01
50	2.11	1.86	1058.50	4	1	2525.91
75	0.47	0.41	1080.17	4	0	2749.93
100	0.21	0.15	1234.33	4	1	2626.93
OVERALL	0.65	0.57	1154.67	19	6	2639.04

Table 4.2. Branch-and-price results for Class 3

the final gap. In particular, given an instance, let UB be the best upper bound found by the beam search. Then, the residual percentage gap can be computed as $\left|\frac{UB-LB_B}{LB_B}\right| \cdot 100$, where LB_B values are those computed when performing the branch-and-price. If the branch-and-price could not find an optimal solution, the beam search might find a better solution. However this is quite rare, as it can be seen in column 5 of Table 4.3. The results show very promising gaps for classes 0, 1, and 2, but not so good for Class 3. This time the high gaps are also justified by the fact that, at the root node, to save time, we *do not* compute the Z_{SC} upper bound which would have improved the accuracy of the method. Of course, increasing the beam size improves the final gap, to the detriment of the computing time. The relative accuracy of the beam search is highly compensated by the small computing time, which is less than 3 minutes, when the branch-and-price requires, on average, up to 45 minutes. Therefore, we can conclude that the proposed beam search is a good compromise between accuracy and computational effort.

4.4.3 VCSBPP comparison

As stated in Chapter 1 and shown in Section 3.1.3, the GBPP generalizes several packing problems, in particular the VCSBPP. Due to its recent introduction, the GBPP literature is quite limited, while for the VCSBPP several heuristic and exact methods are available. In this section, we use the proposed branch-and-price and beam search algorithms to address the VCSBPP and compare the results with those of the state-of-the-art methods specifically designed for the VCSBPP, in particular BB_{HS}, the branch and bound presented in Haouari and Serairi [2011] and VNS_{HSB}, the VNS introduced in Hemmelmayr et al. [2012]. For the beam search, we consider the setting with beam size equal to 4. We consider the instance set of Monaci [2002], which was also used by Haouari and Serairi [2011] and by Hemmelmayr et al. [2012]. Other available VCSBPP instances (see, e.g., Alves and Valério de Carvalho [2007]) do not seem to be sufficiently challenging, as both the branch-and-price and the beam search are able to solve them to optimality at the root node with a negligible computational time.

Table 4.4 compares BB_{HS} with our branch-and-price. The table reports the number of items in the instances and, for each method, the mean percentage gap

CLASS	BEAM	% GAP	OPT	IMPROVING	TIME
0	1	0.33	130	2	23.35
	2	0.29	150	3	28.59
	3	0.28	159	3	31.12
	4	0.26	170	3	33.94
		0.29	**176**	**4**	**29.25**
1	1	1.25	99	3	39.29
	2	1.16	109	3	54.10
	3	1.10	114	2	59.71
	4	0.98	124	2	64.58
		1.12	**128**	**3**	**54.42**
2	1	0.93	103	4	42.43
	2	0.84	113	3	53.64
	3	0.79	119	2	60.22
	4	0.74	123	2	65.89
		0.83	**129**	**4**	**55.54**
3	1	4.97	7	1	145.74
	2	4.72	9	0	155.54
	3	4.70	11	1	157.95
	4	4.68	11	1	158.63
		4.77	**11**	**2**	**154.47**
OVERALL		**1.75**	**444**	**13**	**73.42**

Table 4.3. Beam search results

between the upper and lower bounds at the root node and the number of instances solved to optimality. BB_{HS} performs better. This is due, as stated by the authors in their paper, to a series of dominance criteria and lower bounds specifically designed for the VCSBPP, which, unfortunately, cannot be extended to the GBPP. For instance, the dominance criteria heavily used the hypothesis that the number of available bins for each type is infinite, which is not the case for the GBPP. As expected, since the GBPP is more general, it loses somewhat in efficiently proving optimality, but preserves excellent performances in terms of gaps. A similar behavior can be observed when comparing VNS_{HSB} and the beam search (Table 4.5). In this case, the gap remains under 0.5%, within a competitive computational effort (about two minutes in the worst case).

	BB$_{HS}$		**B&P**	
ITEMS	**% GAP**	**OPT**	**% GAP**	**OPT**
25	0	60	0	60
50	0.01	59	0	60
100	0.02	59	0.1	57
200	0	60	0.6	41
500	0	60	0.11	29

Table 4.4. VCSBPP results: comparison between BB_{HS} and branch-and-price

	VNS$_{HSB}$			**BEAM**		
ITEMS	**% GAP**	**OPT**	**TIME**	**% GAP**	**OPT**	**TIME**
25	0.00	60	150	0.09	54	0.10
50	0.01	59	150	0.21	45	0.53
100	0.00	58	150	0.32	35	3.60
200	0.01	54	150	0.28	20	37.03
500	0.01	52	150	0.41	22	128.44

Table 4.5. VCSBPP results: comparison between VNS_{HSB} and beam search

Chapter 5

The Stochastic Generalized Bin Packing Problem

5.1 Introduction

In this chapter, we present the S-GBPP, a variant of the GBPP where the items are characterized by volume and random profit, and, as for the GBPP, the bins are characterized by volume and cost. Aim of the S-GBPP is to choose a subset of items to be loaded into a subset of bins in order to maximize the expected total net profit, given by the difference between the expected total profit of the loaded items and the total cost of the used bins, while satisfying the volume and bin availability constraints.

In contrast to the deterministic GBPP where we minimize the total net cost, here, in the S-GBPP, we maximize the expected total net profit in order to use, at a later stage, the cumulative left distribution function, which is a standard convention. Vice versa, if we minimized the expected total net cost, then we should use the cumulative right distribution function which, however, is not very common.

The item profits, which also depend on bins where the items will be loaded, are random variables with unknown probability distribution. They are composed by a deterministic profit plus a random term, which represents the profit oscillations due to the handling operations needed for loading the items into the bins.

The S-GBPP frequently arises in real-life applications, in particular in logistics,

where the freight consolidation is essential to optimize the delivery process. In this case, a series of handling operations for bin loading must be performed at the logistic platforms and these operations could significantly affect the final total profit of the loading Tadei et al. [2002].

In this chapter, we introduce a stochastic model of the S-GBPP. In most papers dealing with uncertainty, the probability distribution of the random variables is given and their expected value can then be calculated. This is not the case of the S-GBPP, where the probability distribution of the random item profit is unknown, because it is difficult to be measured in practice and any assumption on its shape would be arbitrary.

We show that, by using some results of the asymptotic theory of extreme values Galambos [1978], the probability distribution of the maximum random profit of any item becomes a Gumbel (or double exponential) probability distribution and the total expected profit of the loaded items can be easily calculated. By using this result a deterministic approximation of the S-GBPP is derived.

This chapter is organized as follows. In Section 5.2, we revisit the assignment model of the GBPP. In particular, we give a more general model which takes general item profits (i.e., depending on the bins into which each item is loaded) into account. This model is the starting point in order to formulate a stochastic model of the S-GBPP, introduced in Section 5.3. Section 5.4 derives the formulation of the probability distribution of the maximum profit of any item, which can be then computed by using the asymptotic approximation introduced in Section 5.5. Section 5.6 gives the deterministic approximation of the original stochastic problem.

5.2 The assignment model of the Generalized Bin Packing Problem revisited

In this section, we provide a more general assignment model of the GBPP. This model is the starting point for the stochastic model of the S-GBPP, introduced in Section 5.3, and takes general item profits into account. In particular, now that profits depend on the bins into which the corresponding items are loaded, we denote by p_{ij} the profit of item $i \in \mathcal{I}$ when loaded into bin $j \in \mathcal{J}$.

With the same notation presented in Section 3.1.1, the assignment model of the GBPP becomes

$$\max_{\{x\},\{y\}} \quad \sum_{i\in\mathcal{I}}\sum_{j\in\mathcal{J}} p_{ij}x_{ij} - \sum_{j\in\mathcal{J}} C_j y_j \tag{5.1}$$

$$\text{subject to} \quad \sum_{j\in\mathcal{J}} x_{ij} \leq 1 \qquad i \in \mathcal{I} \tag{5.2}$$

$$\sum_{i\in\mathcal{I}} w_i x_{ij} \leq W_j \qquad j \in \mathcal{J} \tag{5.3}$$

$$\sum_{i\in\mathcal{I}} x_{ij} \leq |\mathcal{I}| y_j \qquad j \in \mathcal{J} \tag{5.4}$$

$$\sum_{j\in\mathcal{J}:\sigma(j)=t} y_j \geq L_t \qquad t \in \mathcal{T} \tag{5.5}$$

$$\sum_{j\in\mathcal{J}:\sigma(j)=t} y_j \leq U_t \qquad t \in \mathcal{T} \tag{5.6}$$

$$\sum_{j\in\mathcal{J}} y_j \leq U \tag{5.7}$$

$$x_{ij} \in \{0,1\} \qquad i \in \mathcal{I}, j \in \mathcal{J} \tag{5.8}$$

$$y_j \in \{0,1\} \qquad j \in \mathcal{J} \tag{5.9}$$

The objective function (5.1) maximizes the total net profit, given by the difference between the total profit of the loaded items and the total cost of the used bins. As stated in Section 5.1, the objective function appears in a maximization form in order to deal with the cumulative left distribution function when deriving the deterministic approximation in the next Sections.

Constraints (5.2) ensure that each item is loaded into one bin at most. Constraints (5.3) limit the bin capacity. Constraints (5.4) prevent to load items into not used bins. Note that, Constraints (5.3) and (5.4) are equivalent to Constraints (3.2) in the original assignment model of the GBPP presented in Section 3.1.2. The reason of this splitting will be clear in Section 5.3, where Constraints (5.3) will be relaxed, according to a Lagrangian relaxation, in order to derive the deterministic approximation of the S-GBPP. Constraints (5.3) prevent us to have variables y_j within the relaxed objective function. This would have not been the case if we had used Constraints (3.2). Constraints (5.5) and (5.6) give bounds to the minimum and

maximum number of used bins per type, respectively, whilst constraint (5.7) limits the total number of used bins, regardless of their type. Finally, (5.8)-(5.9) are the integrality constraints.

5.3 The Stochastic Generalized Bin Packing Problem

In the S-GBPP the item profit of the GBPP becomes a random variable. In fact, it is composed by a deterministic profit (the one of the GBPP) plus a random term, which represents the profit oscillations due to the handling costs for loading items into bins. We assume that such profit oscillations randomly depend on the handling scenarios which are adopted for bin loading. These random profit oscillations are very difficult to be measured in practice, so that their probability distribution must be assumed as unknown.

The data and variables of the S-GBPP are the same of the GBPP, but some new data and variables must be considered as follows

- S: set of handling scenarios for bin loading
- θ^{jl}: random profit oscillation of loading bin j under handling scenario $l \in S$.

Let us assume, as it is usually done in this context, that θ^{jl} are independent and identically distributed (i.i.d.) random variables with a common probability distribution

$$F(x) = Pr\{\theta^{jl} \leq x\} \tag{5.10}$$

The main feature of our approach consists, as stated above, in considering the probability distribution $F(x)$ as unknown.

Without losing in generality, the random variables θ^{jl} can be scaled by a constant a as follows

$$\tilde{\theta}^{jl} = \theta^{jl} - a \qquad j \in \mathcal{J}, l \in S \tag{5.11}$$

The probability distribution of $\tilde{\theta}^{jl}$ then becomes

$$Pr\{\tilde{\theta}^{jl} \leq x\} = Pr\{\theta^{jl} - a \leq x\} = Pr\{\theta^{jl} \leq x + a\} = F(x + a) \tag{5.12}$$

Let $\tilde{p}_{ij}(\tilde{\theta}^{jl})$ be the random profit of loading item i into bin j under handling scenario l given by

$$\tilde{p}_{ij}(\tilde{\theta}^{jl}) = p_{ij} + \tilde{\theta}^{jl} \qquad i \in \mathcal{I}, j \in \mathcal{J}, l \in S \tag{5.13}$$

Let us define with $\overline{\theta}^{j}$ the maximum of the random profit oscillations of loading bin j among the alternative handling scenarios $l \in S$

$$\overline{\theta}^{j} = \max_{l \in S} \tilde{\theta}^{jl} \qquad j \in \mathcal{J} \tag{5.14}$$

Clearly, $\overline{\theta}^{j}$ is still a random variable with unknown probability distribution given by

$$B_j(x) = \Pr\left\{\overline{\theta}^{j} \leq x\right\} \qquad j \in \mathcal{J} \tag{5.15}$$

As, for any bin j, $\overline{\theta}^{j} \leq x \iff \tilde{\theta}^{jl} \leq x$, $l \in S$ and $\tilde{\theta}^{jl}$ are independent, using (5.12) one gets

$$B_j(x) = \prod_{l \in S} \Pr\left\{\tilde{\theta}^{jl} \leq x\right\} = \prod_{l \in S} F(x+a) = [F(x+a)]^{|S|} \qquad j \in \mathcal{J} \tag{5.16}$$

We assume that the bin loading policy is efficiency-based so that, for any item i and bin j, among the alternative handling scenarios $l \in S$, the one which maximizes the random profit $\tilde{p}_{ij}(\tilde{\theta}^{jl})$ will be selected.

Then, the random profit of loading item i into bin j becomes

$$\overline{p}_{ij}(\overline{\theta}^{j}) = \max_{l \in S} \tilde{p}_{ij}(\tilde{\theta}^{jl}) = p_{ij} + \max_{l \in S} \tilde{\theta}^{jl} = p_{ij} + \overline{\theta}^{j} \qquad i \in \mathcal{I}, j \in \mathcal{J} \tag{5.17}$$

The S-GBPP can be formulated as follows

$$\max_{\{y\}} \quad \left\{ \mathbb{E}_{\{\bar{\theta}\}} \left[\max_{\{x\}} \sum_{i \in \mathcal{I}} \sum_{j \in \mathcal{J}} \overline{p}_{ij}(\overline{\theta}^j) x_{ij} \right] - \sum_{j \in \mathcal{J}} C_j y_j \right\} \tag{5.18}$$

$$\text{subject to} \quad (5.2) - (5.9) \tag{5.19}$$

The objective function (5.18) maximizes the expected total net profit, given by the difference between the expected total profit of the loaded items and the total cost of the used bins.

Let us consider the Lagrangian relaxation of problem (5.18)-(5.19), obtained by relaxing the capacity constraints (5.3) by means of the non negative multipliers $\mu_j, j \in \mathcal{J}$

$$\min_{\{\mu \geq 0\}} \max_{\{y\}} \quad \left\{ \mathbb{E}_{\{\bar{\theta}\}} \left[\max_{\{x\}} \sum_{i \in \mathcal{I}} \sum_{j \in \mathcal{J}} \max \left(0, \overline{p}_{ij}(\overline{\theta}^j) - \mu_j w_i\right) x_{ij} \right] - \sum_{j \in \mathcal{J}} (C_j y_j - \mu_j W_j) \right\} \tag{5.20}$$

$$\text{subject to} \quad (5.2) \text{ and } (5.4) - (5.9) \tag{5.21}$$

The term $max\left(0, \overline{p}_{ij}(\overline{\theta}^j) - \mu_j w_i\right)$ in (5.20) is named the shadow random profit of loading item i into bin j, due to the presence of the shadow prices μ_j.

Problem (5.20)-(5.21) gives an upper bound on the optimal value of problem (5.18)-(5.19), but we know that, when the strong duality conditions are satisfied, the two problems are equivalent.

For any item i, let us consider bin $j = i^*$ (for the sake of simplicity, we assume it is unique), which gives the maximum shadow random profit for item i over all the bins. This quantity represents an upper bound of the actual maximum shadow random profit of item i and becomes

$$\overline{p}_i(\overline{\theta}^{i^*}) - \mu_{i^*} w_i = \max_{j \in \mathcal{J}} \left[\max \left(0, \overline{p}_{ij}(\overline{\theta}^j) - \mu_j w_i\right) \right] \qquad i \in \mathcal{I} \tag{5.22}$$

If item i is loaded, it will select bin j which maximizes its shadow random profit, i.e.

$$x_{ij} = \begin{cases} 1, & \text{if } \; j = i^* \; \text{ and } \; \overline{p}_i(\overline{\theta}^{i^*}) - \mu_{i^*} w_i > 0 \\ 0, & \text{otherwise} \end{cases} \tag{5.23}$$

Using (5.22), (5.23), and the linearity of the expected value operator $\mathbb{E}$, a valid upper bound of the objective function (5.20) becomes

$$\begin{aligned} &\min_{\{\mu \geq 0\}} \max_{\{y\}} \left\{ \sum_{i \in \mathcal{I}} \mathbb{E}_{\{\overline{\theta}^*\}} \left[\overline{p}_i(\overline{\theta}^{i^*}) - \mu_{i^*} w_i \right] - \sum_{j \in \mathcal{J}} (C_j y_j - \mu_j W_j) \right\} = \\ &= \min_{\{\mu \geq 0\}} \max_{\{y\}} \left\{ \sum_{i \in \mathcal{I}} \hat{p}_i - \sum_{j \in \mathcal{J}} (C_j y_j - \mu_j W_j) \right\} \end{aligned} \tag{5.24}$$

where

$$\hat{p}_i = \mathbb{E}_{\{\overline{\theta}^*\}} \left[\overline{p}_i(\overline{\theta}^{i^*}) - \mu_{i^*} w_i \right] \qquad i \in \mathcal{I} \tag{5.25}$$

The calculation of $\hat{p}_i$ in (5.25) requires to know the probability distribution of the maximum shadow random profit of item i, i.e. $\overline{p}_i(\overline{\theta}^{i^*}) - \mu_{i^*} w_i$, which will be introduced in the next section.

5.4 Formulation of the probability distribution of the maximum shadow random profit of any item

By (5.17) and (5.22), let

$$G_i(x) = Pr\left\{ \overline{p}_i(\overline{\theta}^{i^*}) - \mu_{i^*} w_i \leq x \right\} = Pr\left\{ \max_{j \in \mathcal{J}} \left[p_{ij} + \overline{\theta}^j - \mu_j w_i \right]^+ \leq x \right\} \qquad i \in \mathcal{I} \tag{5.26}$$

be the probability distribution of the maximum shadow random profit of item i.

As, for any item i, $\max_{j \in \mathcal{J}} \left[p_{ij} + \overline{\theta}^j - \mu_j w_i \right]^+ \leq x \iff \left[p_{ij} + \overline{\theta}^j - \mu_j w_i \right]^+ \leq$

$x, \quad j \in \mathcal{J}$, and the random variables $\overline{\theta}^j$ are independent (because θ^{jl} are independent), due to (5.15) and (5.16), $G_i(x)$ in (5.26) becomes a function of the number $|S|$ of handling scenarios for bin loading

$$\begin{aligned} G_i(x,|S|) &= Pr\left\{\max_{j\in\mathcal{J}}\left[p_{ij}+\overline{\theta}^j-\mu_j w_i\right]^+ \le x\right\} = \\ &= \prod_{j\in\mathcal{J}} Pr\left\{\left[p_{ij}+\overline{\theta}^j-\mu_j w_i\right]^+ \le x\right\} = \\ &= \prod_{j\in\mathcal{J}:p_{ij}+\overline{\theta}^j-\mu_j w_i>0} Pr\left\{\overline{\theta}^j \le x-p_{ij}+\mu_j w_i\right\} = \\ &= \prod_{j\in\mathcal{J}:p_{ij}+\overline{\theta}^j-\mu_j w_i>0} B_j\left(x-p_{ij}+\mu_j w_i\right) = \\ &= \prod_{j\in\mathcal{J}:p_{ij}+\overline{\theta}^j-\mu_j w_i>0} \left[F\left(x-p_{ij}+\mu_j w_i+a\right)\right]^{|S|} \qquad i\in\mathcal{I} \end{aligned} \tag{5.27}$$

Let us now set the value of the constant a in (5.27) equal to the root of the equation

$$1-F(a)=1/|S| \tag{5.28}$$

Let us assume that $|S|$ is large enough to use the asymptotic approximation $\lim_{|S|\to+\infty} G_i(x,|S|)$ as a good approximation of $G_i(x)$, i.e.

$$G_i(x)=\lim_{|S|\to+\infty} G_i(x,|S|)) \qquad i\in\mathcal{I} \tag{5.29}$$

The calculation of the limit in (5.29) would require to know the shape of the probability distribution $F(.)$, which is still unknown. To overcome this problem, in the next Section we will show that under a mild assumption on the shape of the unknown probability distribution $F(.)$, the limit in (5.29) will tend towards a specific functional form.

5.5 The asymptotic approximation of the probability distribution of the maximum shadow random profit of any item

The method we use to calculate the asymptotic approximation of the probability distribution of the maximum shadow random profit of item i, $G_i(x)$, derives from the asymptotic theory of extreme values Galambos [1978] and is based on the following observation.

Under the assumption that $F(.)$ is asymptotically exponential in its right tail, i.e. there is a constant $\beta > 0$ such that

$$\lim_{z \to +\infty} \frac{1 - F(x+z)}{1 - F(z)} = e^{-\beta x} \tag{5.30}$$

the limit in (5.29) tends towards a specific functional form, which is a Gumbel (or double exponential) probability distribution Gumbel [1958].

(5.30) is a very mild assumption for the unknown probability distribution $F(x)$ as we observe that many probability distributions show such behavior, among them the widely used distributions Exponential, Normal, Lognormal, Gamma, Gumbel, Laplace, and Logistic.

Let us consider the following theorem, which provides the desired asymptotic approximation for $G_i(x)$.

Theorem 2. *Under assumption (5.30), the asymptotic approximation of the probability distribution $G_i(x), i \in \mathcal{I}$, becomes the following Gumbel probability distribution*

$$G_i(x) = \lim_{|S| \to +\infty} G_i(x, |S|)) = \exp\left(-A_i e^{-\beta x}\right) \qquad i \in \mathcal{I}, \tag{5.31}$$

where

$$A_i = \sum_{j \in \mathcal{J}: p_{ij} + \overline{\theta}^j - \mu_j w_i > 0} e^{\beta(p_{ij} - \mu_j w_i)} \qquad i \in \mathcal{I} \tag{5.32}$$

is the "accessibility", in the sense of Hansen [Hansen, 1959], of item i to the set of bins.

Proof. By (5.27) and (5.29) one has

$$\begin{aligned} G_i(x) &= \lim_{|S|\to+\infty} G_i(x,|S|)) = \lim_{|S|\to+\infty} \prod_{j\in\mathcal{J}:p_{ij}+\overline{\theta}^j-\mu_j w_i>0} \left[F\left(x-p_{ij}+\mu_j w_i+a\right)\right]^{|S|} = \\ &= \prod_{j\in\mathcal{J}:p_{ij}+\overline{\theta}^j-\mu_j w_i>0} \lim_{|S|\to+\infty} \left[F\left(x-p_{ij}+\mu_j w_i+a\right)\right]^{|S|} \end{aligned} \tag{5.33}$$

Let us consider in (5.33)

$$\lim_{|S|\to+\infty} F\left(x-p_{ij}+\mu_j w_i+a\right) \tag{5.34}$$

As by (5.28) $\lim_{|S|\to+\infty} a = +\infty$ (because $F(a)$ tends to 1), from (5.30) one obtains

$$\lim_{|S|\to+\infty} \frac{1-F(x-p_{ij}+\mu_j w_i+a)}{1-F(a)} = e^{-\beta(x-p_{ij}+\mu_j w_i)} \tag{5.35}$$

Then, by (5.28) and (5.35), (5.34) becomes

$$\begin{aligned} \lim_{|S|\to+\infty} F(x-p_{ij}+\mu_j w_i+a) &= \lim_{|S|\to+\infty} \left(1-[1-F(a)]e^{-\beta(x-p_{ij}+\mu_j w_i)}\right) = \\ &= \lim_{|S|\to+\infty} \left(1+\frac{-e^{-\beta(x-p_{ij}+\mu_j w_i)}}{|S|}\right) \end{aligned} \tag{5.36}$$

Substituting (5.36) into (5.33) one gets

$$G_i(x) = \prod_{j\in\mathcal{J}:p_{ij}+\overline{\theta}^j-\mu_j w_i>0} \lim_{|S|\to+\infty} \left(1+\frac{-e^{-\beta(x-p_{ij}+\mu_j w_i)}}{|S|}\right)^{|S|} \tag{5.37}$$

and, by reminding that $\lim_{n\to+\infty}(1+\frac{x}{n})^n = \exp(x)$, using (5.32) one finally gets

$$G_i(x) = \prod_{j\in\mathcal{J}:p_{ij}+\overline{\theta}^j-\mu_j w_i>0} \exp\left(-e^{-\beta(x-p_{ij}+\mu_j w_i)}\right) = \exp\left(-A_i e^{-\beta x}\right) \tag{5.38}$$

□

5.6 The deterministic approximation of the S-GBPP

Using the probability distribution $G_i(x)$ given by (5.31), we are now able to calculate $\hat{p}_i$ in (5.25) as follows

$$\hat{p}_i = \int_{-\infty}^{+\infty} x dG_i(x) = \int_{-\infty}^{+\infty} x \exp\left(-A_i e^{-\beta x}\right) A_i e^{-\beta x} \beta dx \qquad i \in \mathcal{I} \tag{5.39}$$

Substituting for $v = A_i e^{-\beta x}$ one gets

$$\begin{aligned} \hat{p}_i &= -1/\beta \int_0^{+\infty} \ln(v/A_i) e^{-v} dv = \\ &= -1/\beta \int_0^{+\infty} e^{-v} \ln v dv + 1/\beta \ln A_i \int_0^{+\infty} e^{-v} dv = \\ &= \gamma/\beta + 1/\beta \ln A_i = \\ &= 1/\beta(\ln A_i + \gamma) \end{aligned} \tag{5.40}$$

where $\gamma = -\int_0^{+\infty} e^{-v} \ln v \; dv \simeq 0.5772$ is the Euler constant.

By (5.40) and but the constant $\frac{\gamma}{\beta}|\mathcal{I}|$, the objective function (5.24) becomes

$$\min_{\{\mu \geq 0\}} \max_{\{y\}} \left\{ 1/\beta \sum_{i \in \mathcal{I}} \ln A_i - \sum_{j \in \mathcal{J}} (C_j y_j - \mu_j W_j) \right\} \tag{5.41}$$

By defining the *total accessibility* of items to the set of bins as

$$\Phi = \prod_{i \in \mathcal{I}} A_i, \tag{5.42}$$

(5.41) becomes

$$\begin{aligned} & \min_{\{\mu \geq 0\}} \max_{\{y\}} \left\{ \frac{1}{\beta} \sum_{i \in \mathcal{I}} \ln A_i - \sum_{j \in \mathcal{J}} (C_j y_j - \mu_j W_j) \right\} = \\ = \; & \min_{\{\mu \geq 0\}} \max_{\{y\}} \left\{ \frac{1}{\beta} \ln \prod_{i \in \mathcal{I}} A_i - \sum_{j \in \mathcal{J}} (C_j y_j - \mu_j W_j) \right\} = \\ = \; & \min_{\{\mu \geq 0\}} \max_{\{y\}} \left\{ \frac{1}{\beta} \ln \Phi - \sum_{j \in \mathcal{J}} (C_j y_j - \mu_j W_j) \right\} \end{aligned} \tag{5.43}$$

which gives the desired deterministic approximation of the upper bound of the S-GBPP.

By comparing (5.43) with (5.24), it is interesting to observe that the total expected shadow profit of the loaded items, $\sum_{i \in \mathcal{I}} \hat{p}_i$, is proportional to the logarithm of the total accessibility of those items to the set of bins Φ.

Let us note that (5.43) requires to know a proper value of the positive constant β, which is the same parameter that appears in the Gumbel probability distribution (5.31). This can be obtained by calibration as follows.

Let us consider the standard Gumbel distribution $G(x) = \exp(-e^{-x})$. If one accepts an approximation error of 0.01, then $G(x) = 1 \iff x = 4.60$ and $G(x) = 0 \iff x = -1.52$.

Let us consider the interval $[m, M]$, where the shadow random profits $\overline{p}_i(\overline{\theta}^{i^*}) - \mu_{i^*} w_i$ are drawn from.

The following equations hold

$$\beta(m - \zeta) = -1.52, \quad \beta(M - \zeta) = 4.60, \tag{5.44}$$

where ζ is the mode of the Gumbel distribution $G(x) = \exp\left(-e^{-\beta(x-\zeta)}\right)$.
From (5.44) one finally gets

$$\beta = \frac{6.12}{M - m} \tag{5.45}$$

Chapter 6

On-line Generalized Bin Packing Problems

6.1 Introduction

In this chapter, we present a detailed study on the on-line generalized bin packing problems: a new family of on-line problems not yet studied in the literature. These problems are the On-line Generalized Bin Packing Problem (OGBPP), the On-line Generalized Bin Packing Problem with item profits proportional to item volumes (OGBPP$_\kappa$), and the On-line Variable Cost and Size Bin Packing Problem (OVCSBPP), and arise in many applications where the orders, represented by the items, arrive on-line in an unpredictable way to a decision maker.

We analyze these problems along a different research direction: we investigate whether the tools used by researchers to qualify on-line algorithms are still effective when applied to richer problems. These tools are the asymptotic and the absolute worst case ratio. We propose a noteworthy number of algorithms and we show that, for each of them, it is impossible to certify their performance when applied to the OGBPP. In particular, we prove a stronger result than the one achieved in the OKP. In fact, as Iwama and Zhang [2007, 2010] showed that the OKP is not competitive (i.e., its absolute worst case ratio is infinite), we prove that, for the proposed algorithms, it is even impossible to apply the definition of these worst case ratios. Moreover, we prove that this behavior also holds for off-line algorithms

like the FFD and the BFD introduced in Section 3.3.1, and even if we consider the maximization form of the OGBPP. The reason why we take into account both forms of the problem is in accordance with previous works in the literature concerning the on-line variant of other problems. For example, in the OKP, Han and Makino [2010] proved that, switching from the maximization to the minimization form of the same problem, the absolute worst case ratio becomes finite. We also show that, even in the particular case of the OGBPP$_\kappa$, the available tools do not allow us to estimate any performance ratio for the proposed algorithms. We believe that the ultimate packing problem for which it is possible to compute a performance ratio is the OVCSBPP. For this problem, we generalize the work of Li and Chen [2006] to a more general setting, still guaranteeing the same performance ratio equal to 2.

This chapter is organized as follows. In Section 6.2, we provide the formal definitions of the asymptotic and absolute worst ratios and describe the problem settings of the OGBPP, the GBPP$_\kappa$ and its on-line variant, the OGBPP$_\kappa$, and the VCSBPP and its on-line variant, the OVCSBPP.

In Section 6.3, we propose both on-line and off-line algorithms for the OGBPP and we show that, for each of them, it is impossible to compute the asymptotic and the absolute worst case ratio. In Section 6.4, we study the particular case of the OGBPP$_\kappa$ where the item profits are proportional to the item volume through a positive proportionality coefficient κ. We show that, even in this particular case, the same conclusions shown in Section 6.3 hold. In Section 6.5, we apply to the OVCSBPP the on-line algorithm First Fit (FF), which, as mentioned in Chapter 2, was initially proposed by Johnson et al. [1974] for the BPP. We prove that the asymptotic worst case ratio of the FF when applied to the OVCSBPP is equal to 2 and that this bound is tight.

6.2 Problems settings

In this section, we first define the most common tools used to estimate algorithms performance: the asymptotic and the absolute worst case ratios. Then, we give a formal definition of the OGBPP, the GBPP$_\kappa$, the OGBPP$_\kappa$, the VCSBPP, and the OVCSBPP.

6.2.1 The asymptotic and the absolute worst case ratios

Several denominations and definitions can be found in the literature for the asymptotic and the absolute worst case ratios. Sometimes they are named *performance ratios* or, particularly in scheduling problems, *competitive ratios*. Roughly speaking, they reveal how far is the solution value found by an algorithm from the optimum in the worst case. Consequently, this estimation is guaranteed for any instance of the problem to which the algorithm is applied. Formally, given a *minimization* problem Π, an instance $I \in \Pi$ of the problem, an algorithm $\mathcal{A}$, the objective function value of the solution yielded by the algorithm $\mathcal{A}(I)$, and the optimum $\mathrm{OPT}(I)$, then *the asymptotic worst case ratio* is the smallest $\mathcal{R}$ such that the following relation between the optimum and the value yielded by the algorithm yields *for any instance* of the problem:

$$\mathcal{A}(I) \leq \mathcal{R} \cdot \mathrm{OPT}(I) + \mathcal{O}(1), \qquad \forall\, I \in \Pi, \tag{6.1}$$

where $\mathcal{O}(1)$ is a constant independent of the instances. The *absolute worst case ratio* is the smallest ρ such that the following relation between the optimum and the value yielded by the algorithm yields *for any instance* of the problem:

$$\mathcal{A}(I) \leq \rho \cdot \mathrm{OPT}(I), \qquad \forall\, I \in \Pi \tag{6.2}$$

The asymptotic and the absolute worst case ratios are also respectively defined, sometimes, in terms of *tight bounds* as follows:

$$\mathcal{R} = \lim_{n \to +\infty} \sup_{I \in \Pi} \left\{ \frac{\mathcal{A}(I)}{\mathrm{OPT}(I)} \;\middle|\; \mathrm{OPT}(I) \geq n \right\} \tag{6.3}$$

$$\rho = \sup_{I \in \Pi} \left\{ \frac{\mathcal{A}(I)}{\mathrm{OPT}(I)} \right\} = \inf \left\{ r \;\middle|\; \frac{\mathcal{A}(I)}{\mathrm{OPT}(I)} \leq r,\ \forall\, I \in \Pi \right\} \tag{6.4}$$

Unfortunately, there is not uniformity of definitions among authors in the literature for the asymptotic and the absolute worst case ratio when applied to a *maximization* problem. In accordance to authors like [Fisher, 1980, Martello and Toth, 1990, Wang and Xing, 2009, Wang, 2012], we believe that the most correct definition for the asymptotic and the absolute worst case ratios applied to a maximization problem

are, respectively, the greatest $\mathcal{R}$ and ρ, such that, for any instance of the problem, we have:

$$\mathcal{A}(I) \geq \mathcal{R} \cdot \mathrm{OPT}(I) + \mathcal{O}(1), \qquad \forall\, I \in \Pi \tag{6.5}$$

$$\mathcal{A}(I) \geq \rho \cdot \mathrm{OPT}(I) \qquad \forall\, I \in \Pi, \tag{6.6}$$

where $\mathcal{O}(1)$ is a constant independent of the instance.

Nevertheless, authors like [Johnson, 1973b, Iwama and Taketomi, 2002, Iwama and Zhang, 2007, 2010] prefer to work with the ratio $\frac{\mathrm{OPT}(I)}{\mathcal{A}(I)}$ rather than $\frac{\mathcal{A}(I)}{\mathrm{OPT}(I)}$. The reason for this choice is that, both for minimization and maximization problems, the computation of any performance ratio reduces to finding the *smallest* value of the ratio itself for any instance of the problem. Throughout this paper we will refer to definitions (6.5) and (6.6) when dealing with a maximization problem.

6.2.2 The On-line Generalized Bin Packing Problem

In this section, we provide a formal definition of the On-line Generalized Bin Packing Problem with item profits proportional to item volumes. The OGBPP, is the on-line variant of the GBPP where the items arrive on-line to a decision maker. When the decision maker receives an item, its information is revealed. Information disclosed on item arrival is: item volume, item profit, whether the item is compulsory or non-compulsory, and whether the received item is the last one. According to the on-line variants of the BPP and of the VSBPP, we assume that the number of available bins for each bin type is unlimited (see, for instance, Friesen and Langston [1986]). Moreover, the case with a limited number of bins is easy to address. In fact, it can be easily reduced to an OKP by setting $|\mathcal{T}| = 1$ (without loss of generality, we can assume $t = 1$ only), $L_1 = U_1 = U = 1$, and $\mathcal{C}_1 = 0$. As stated in Section 6.1, Iwama and Zhang [2007, 2010] proved that the OKP is not competitive.

6.2.3 The Generalized Bin Packing Problem with item profits proportional to item volumes and its on-line variant

In this section, we provide a formal definition of the Generalized Bin Packing Problem with item profits proportional to item volumes (GBPP_κ) with its on-line variant, the On-line Generalized Bin Packing Problem with item profits proportional to item volumes (OGBPP_κ). The GBPP_κ is a particular case of the GBPP where item profits are proportional to item volumes through a positive constant κ:

$$p_i = \kappa\, w_i, \qquad \forall i \in \mathcal{I}. \tag{6.7}$$

This problem arises in all those applications where the cost for shipping an item is proportional to the size of the item.

The OGBPP_κ, is the on-line variant of the GBPP_κ where the items arrive on-line to a decision maker. When the decision maker receives an item, its information is revealed. Information disclosed on item arrival is: item volume (item profit is consequently computed multiplying item volume by κ, which is known a priori), whether the item is compulsory or non-compulsory, and whether the received item is the last one. As for the OGBPP, we study the particular case with an unlimited supply of bins. In fact, as seen in Section 6.2.2, the case with a limited number of bins can be easily reduced to the OKP which, as proven by [Iwama and Zhang, 2007, 2010], is not competitive.

6.2.4 The Variable Cost and Size Bin Packing Problem and its on-line variant

In this section, we provide a formal definition of the Variable Cost and Size Bin Packing Problem (VCSBPP) with its on-line variant, the On-line Variable Cost and Size Bin Packing Problem (OVCSBPP). Although the VCSBPP has already been described in Chapter 2 and in Section 3.1.3, for the sake of clarity we provide here a more formal definition of the problem. In the VCSBPP, a set of items $\mathcal{I}$, with $|\mathcal{I}| = n$ has to be accommodated into bins, each characterized by volume (or capacity) and cost. The goal is to find the best assignment among items and bins

in order to minimize the overall cost, given by the sum of the costs of the selected bins. The bins are classified into types and, as for the GBPP, the set of bin types is denoted by $\mathcal{T}$. Each item $i \in \mathcal{I}$ is characterized by a volume w_i and each bin of type $t \in \mathcal{T}$ is characterized by a cost $\mathcal{C}_t$ and by a capacity $\mathcal{W}_t$.

In the on-line variant of the problem, the OVCSBPP, the items arrive on-line to a decision maker. When the decision maker receives an item, its information is revealed. Information disclosed on item arrival is item volume and whether the received item is the last one.

6.2.5 Terminology

Throughout this chapter, we introduce, in addition to that already presented in Section 3.1.1, the following terminology. Given any instance I of problem Π, with $\Pi \in \{\text{VCSBPP, OVCSBPP, GBPP, OGBPP}, \text{GBPP}_\kappa, \text{OGBPP}_\kappa\}$, we name

- $B \subset \mathcal{J}$ the set of bins selected by any algorithm applied to instance $I \in \Pi$, with $p = |B|$
- $B^* \subset \mathcal{J}$ the set of bins of an optimal solution of instance $I \in \Pi$, with $g = |B^*|$
- $\beta(j)$ the *level* of bin $j \in B \cup B^*$, i.e., the sum of the volumes of the items loaded into bin j
- s the bin type with the least cost over volume ratio, i.e., $s = \arg\min_{t \in \mathcal{T}} \frac{\mathcal{C}_t}{\mathcal{W}_t}$.

Moreover, according to the bin packing literature, we name *open* bins all those bins containing items at a given moment.

6.3 Algorithms for the On-line Generalized Bin Packing Problem

In this section, we present a wide variety of algorithms for the OGBPP. We start extending to the OGBPP the FF algorithm introduced by Johnson et al. [1974] for the BPP. We also propose and study advanced variants based on this algorithm. These variants are the First Fit with Rejections (FFR) and the Best Fit

WITH REJECTIONS (BFR), where non-profitable bins are rejected at the end of the process, the FIRST FIT WITH BUFFER AND REJECTIONS (FFBR), where the arrived items are stored into a buffer with a limited size before loading them into any bin, and the FIRST FIT WITH TIME BUFFER AND REJECTIONS (FFTBR), where the decision maker stores the received items into an unlimited buffer and periodically load them into bins. We prove that for the FF algorithm and for all its variants the most popular tools used to analyze algorithm performance cannot be applied. In particular, we prove that it is impossible to compute both the asymptotic and the absolute worst case ratios. This drawback also holds for off-line algorithms such as the well known FFD and BFD. Moreover, we prove that the same issue is encountered even when dealing with the maximization form of the problem.

The FF algorithm works as follows. Every time an item $i \in \mathcal{I}$ arrives on-line, we try to load it into the first bin (among the already open ones) able to contain it. If none among the open bins is able to accommodate item $i \in \mathcal{I}$, then we select a new bin. Whilst in the BPP all the bins are equal, here, since bins are classified into types, several choices are possible. A possible choice, which is the one we will make when addressing the OVCSBPP in Section 6.5, is to select a new bin of type s, defined as in Section 6.2.5. Note that, this definition also generalizes the FF introduced by Johnson et al. [1974] for the BPP. If fact, in this particular case, bin type s coincides with the only bin type in the BPP. However, as we show in the next theorems, even with instances of the OGBPP with just one bin type it is impossible to apply the definition of the worst case ratios defined in Section 6.2.1. Therefore, even selecting the next new bin according to a different criterion would lead to the same conclusion. A variant of the FF algorithm is the BEST FIT (BF) algorithm which works like the former with the only difference that, as for the BFD presented in Section 3.3.1, it tries to accommodate a new item $i \in \mathcal{I}$ into the *best* bin; the one with the least residual space after placing, if possible, item $i \in \mathcal{I}$ into it. In Theorem 3, we show that it is impossible to compute the asymptotic and the absolute worst case ratio for both FF and BF algorithms when applied to the OGBPP.

Theorem 3. *It is impossible to compute the asymptotic and the absolute worst case ratio for the* FF *and* BF *algorithms for both forms of the* OGBPP.

Proof. Consider instance $I(\nu_A, \nu_B, \nu_C)$, composed by one bin type ($|\mathcal{T}| = 1$ and $t = 1$) with $\mathcal{W}_1 = \mathcal{W}$, $\mathcal{C}_1 = \mathcal{C}$, ν_A type A compulsory items, ν_B type B non-compulsory items, and ν_C type C non-compulsory items. Let $w_A = \mathcal{W}$, $w_B = \mathcal{W}$, $p_B = \mathcal{C} + \epsilon$, $w_C = \mathcal{W}$, and $p_C = \mathcal{C} - \epsilon$, with $\epsilon > 0$. Since all the items have volume equal to $\mathcal{W}$, bins can accommodate only one item. Consequently, we must employ ν_A bins to accommodate all the ν_A type A compulsory items and ν_B bins to accommodate all the ν_B type B non-compulsory items, which are taken because $p_B > \mathcal{C}$. Since no type C item is profitable and cannot be loaded with any other item, the optimal solution will not contain any of them. Considering the minimization form of the problem, we have:

$$\mathrm{OPT}(I(\nu_A, \nu_B, \nu_C)) = \nu_A\mathcal{C} + \nu_B(\mathcal{C} - p_B) = \nu_A\mathcal{C} - \nu_B\epsilon \tag{6.8}$$

Consider the following possible on-line sequence of items:

$$\overbrace{A \quad \ldots \quad A}^{\nu_A \text{ times}} \quad \overbrace{B \quad \ldots \quad B}^{\nu_B \text{ times}} \quad \overbrace{C \quad \ldots \quad C}^{\nu_C \text{ times}}$$

Applying either FF or BF to the above sequence, we have:

$$\begin{aligned} \mathrm{FF}(I(\nu_A, \nu_B, \nu_C)) = \mathrm{BF}(I(\nu_A, \nu_B, \nu_C)) &= \nu_A\mathcal{C} + \nu_B(\mathcal{C} - p_B) + \nu_C(\mathcal{C} - p_C) \\ &= \nu_A\mathcal{C} - \nu_B\epsilon + \nu_C\epsilon \end{aligned} \tag{6.9}$$

According to (6.1), in order to compute the asymptotic worst case ratio, we have to find a proper constant $\mathcal{O}(1)$ and the least $\mathcal{R}$, both independent of the instances, such that

$$\nu_A\mathcal{C} - \nu_B\epsilon + \nu_C\epsilon \leq \mathcal{R}(\nu_A\mathcal{C} - \nu_B\epsilon) + \mathcal{O}(1) \tag{6.10}$$

If we consider instance $I(0, 0, \nu_C)$, (6.10) becomes

$$\nu_C\epsilon \leq \mathcal{O}(1), \tag{6.11}$$

which is impossible because a constant cannot be greater than a linear ($\mathcal{O}(\nu)$) term.

According to (6.2), in order to compute the absolute worst case ratio, we have to find the least ρ, independent by the instances, such that

$$\nu_A\mathcal{C} - \nu_B\epsilon + \nu_C\epsilon \leq \rho(\nu_A\mathcal{C} - \nu_B\epsilon) \tag{6.12}$$

If we consider instance $I(1,\, 0,\, \nu_C)$, (6.12) becomes

$$\mathcal{C} + \nu_C\epsilon \leq \rho\mathcal{C}, \tag{6.13}$$

which implies $\rho \to +\infty$, since ν_C can be arbitrarily large. On the contrary, if we consider instance $I(0,\, 1,\, \nu_C)$, (6.12) becomes

$$-\epsilon + \nu_C\epsilon \leq -\rho\epsilon, \tag{6.14}$$

which implies $\rho \to -\infty$, since ν_C can be arbitrarily large. But this contradicts $\rho \to +\infty$ in order to satisfy (6.13). Therefore it is impossible even to compute the absolute worst case ratio.

Considering the maximization form of the problem we have

$$\mathrm{OPT}(I(\nu_A,\, \nu_B,\, \nu_C)) = \nu_B(p_B - \mathcal{C}) - \nu_A\mathcal{C} = -\nu_A\mathcal{C} + \nu_B\epsilon \tag{6.15}$$

and

$$\begin{aligned} \mathrm{FF}(I(\nu_A,\, \nu_B,\, \nu_C)) = \mathrm{BF}(I(\nu_A,\, \nu_B,\, \nu_C)) &= -\nu_A\mathcal{C} + \nu_B(p_B - \mathcal{C}) + \nu_C(p_C - \mathcal{C}) \\ &= -\nu_A\mathcal{C} + \nu_B\epsilon - \nu_C\epsilon \end{aligned} \tag{6.16}$$

According to (6.5), in order to compute the asymptotic worst case ratio, we have to find a proper constant $\mathcal{O}(1)$ and the greatest $\mathcal{R}$, both independent of the instances, such that

$$-\nu_A\mathcal{C} + \nu_B\epsilon - \nu_C\epsilon \geq \mathcal{R}(-\nu_A\mathcal{C} + \nu_B\epsilon) + \mathcal{O}(1) \tag{6.17}$$

If we consider instance $I(0,\, 0,\, \nu_C)$, (6.17) becomes

$$-\nu_C\epsilon \geq \mathcal{O}(1), \tag{6.18}$$

which is impossible because a constant cannot be less than a negative linear (- $\mathcal{O}(\nu)$) term.

According to (6.6), in order to compute the absolute worst case ratio, we have to find the greatest ρ, independent by the instances, such that

$$-\nu_A\mathcal{C} + \nu_B\epsilon - \nu_C\epsilon \geq \rho(-\nu_A\mathcal{C} + \nu_B\epsilon) \tag{6.19}$$

If we consider instance $I(1,\, 0,\, \nu_C)$, (6.19) becomes

$$-\nu_C\epsilon - \mathcal{C} \geq -\rho\mathcal{C}, \tag{6.20}$$

which implies $\rho \to +\infty$, since ν_C can be arbitrarily large. On the contrary, if we consider instance $I(0,\, 1,\, \nu_C)$, (6.19) becomes

$$-\nu_C\epsilon + \epsilon \geq \rho\epsilon, \tag{6.21}$$

which implies $\rho \to -\infty$, since ν_C can be arbitrarily large. But this contradicts $\rho \to +\infty$ in order to satisfy (6.20). □

We also want to point out that, if one used definitions (6.3) and (6.4) to compute the asymptotic and the worst case ratios, one respectively would find (just for instances $I(\nu_A,\, \nu_B,\, \nu_C)$) $\mathcal{R}(I) = 1 + \frac{\epsilon}{\mathcal{C}}$ and $\rho(I) \to +\infty$. However, definitions (6.3) and (6.4) cannot be used for this problem because they implicitly assume that $\mathrm{OPT}(I)$ is a positive number. This implicit assumption becomes explicit considering that

$$\mathcal{A}(I) \leq \mathcal{R} \cdot \mathrm{OPT}(I) + \mathcal{O}(1), \qquad \forall\, I \in \Pi \tag{6.22}$$

Dividing by $\mathrm{OPT}(I)$, we get

$$\frac{\mathcal{A}(I)}{\mathrm{OPT}(I)} \leq \mathcal{R} + \frac{\mathcal{O}(1)}{\mathrm{OPT}(I)}, \qquad \forall\, I \in \Pi \tag{6.23}$$

However, this division is allowed only if $\mathrm{OPT}(I) > 0$ *for all* the instances I of problem Π. Otherwise, either the division is impossible (when $\mathrm{OPT}(I) = 0$), either

the inequality is violated (when $\mathrm{OPT}(I) < 0$). Note that, when $\mathrm{OPT}(I) \to +\infty$, (6.23) becomes

$$\mathcal{R} \geq \lim_{\mathrm{OPT}(I)\to+\infty} \frac{\mathcal{A}(I)}{\mathrm{OPT}(I)}, \qquad \forall\, I \in \Pi. \tag{6.24}$$

Since this inequality must hold for all the instances I of problem Π, we should consider the upper extreme of the ratio, but this is equivalent to definition (6.3) which, however, cannot be applied because, as shown in Theorem 3, $\mathrm{OPT}(I)$ can also assume null and negative values. A similar conclusion holds for definition (6.4) concerning the absolute worst case ratio.

Before proposing more sophisticated on-line algorithms, we prove a lemma to which we will refer within the proofs of the next theorems. In particular, we show that, due to the presence of non-compulsory items, there exist instances where the optimum and the value yielded by an on-line algorithm differ by a linear term which, in principle, can be arbitrarily large. In Lemma 1, we show that it is impossible to guarantee both an absolute and an asymptotic worst case ratio in these circumstances.

Lemma 1. *Given a minimization problem* Π *and an algorithm* $\mathcal{A}$*, let* $I(\mu, \nu) \in \Pi$ *be an instance with* $\mu, \nu \in \mathbb{N}$*, such that* $\mathcal{A}(I(\mu, \nu)) = \alpha\mu$*, and* $\mathrm{OPT}(I(\mu, \nu)) = \beta\mu - \gamma\nu$*, with* $\alpha, \beta, \gamma > 0$*. Then, it is impossible to compute the asymptotic and the absolute worst case ratio for algorithm* $\mathcal{A}$*, even in the maximization form of problem* Π*.*

Proof. We first prove the lemma considering problem Π in its minimization form. If there exists an asymptotic worst case ratio then, according to (6.1), we have to find proper $\mathcal{R}$ and $\mathcal{O}(1)$ such that

$$\alpha\mu \leq \mathcal{R}(\beta\mu - \gamma\nu) + \mathcal{O}(1). \tag{6.25}$$

If we consider the particular instance $I(0, \nu)$, then (6.25) becomes

$$0 \leq -\mathcal{R}\gamma\nu + \mathcal{O}(1). \tag{6.26}$$

Since ν can be arbitrarily large and $\gamma > 0$, then, independently of the constant $\mathcal{O}(1)$, it must be $\mathcal{R} \leq 0$. Vice versa, considering instance $I(\mu, 0)$, (6.25) becomes

$$\alpha\mu \leq \mathcal{R}\beta\mu + \mathcal{O}(1). \tag{6.27}$$

Since μ can be arbitrarily large and $\beta > 0$, then, independently of the constant $\mathcal{O}(1)$, it must be $\mathcal{R} \geq \frac{\alpha}{\beta}$. Since, by hypothesis, both α and β are positive numbers, then their ratio is a positive number. Hence requiring $\mathcal{R} \geq \frac{\alpha}{\beta}$ contradicts the previous requirement that $\mathcal{R} \leq 0$. Therefore, it is impossible to compute the asymptotic worst case ratio. Since this result holds independently of constant $\mathcal{O}(1)$ and, according to (6.1) and (6.2), the absolute worst case ratio is the particular case when $\mathcal{O}(1) = 0$, then it is also impossible to compute the absolute worst case ratio.

If we consider the maximization form of problem Π, we have that $\mathcal{A}(I(\mu, \nu)) = -\alpha\mu$, and $\mathrm{OPT}(\mu, \nu) = \gamma\nu - \beta\mu$. According to (6.5), we have to find proper $\mathcal{R}$ and $\mathcal{O}(1)$ such that

$$-\alpha\mu \geq \mathcal{R}(\gamma\nu - \beta\mu) + \mathcal{O}(1). \tag{6.28}$$

If we consider the particular instance $I(0, \nu)$, then (6.28) becomes

$$0 \geq \mathcal{R}\gamma\nu + \mathcal{O}(1). \tag{6.29}$$

Since ν can be arbitrarily large and $\gamma > 0$, then, independently of the constant $\mathcal{O}(1)$, it must be $\mathcal{R} \leq 0$. Vice versa, considering instance $I(\mu, 0)$, (6.28) becomes

$$-\alpha\mu \geq -\mathcal{R}\beta\mu + \mathcal{O}(1). \tag{6.30}$$

Since μ can be arbitrarily large and $\beta > 0$, the, independently of the constant $\mathcal{O}(1)$, it must be $\mathcal{R} \geq \frac{\alpha}{\beta}$. Since, by hypothesis, both α and β are positive numbers, then their ratio is a positive number and we get to the same conclusion of the minimization form of the problem. □

A natural improvement of FF and BF would be, at the end of the process, to reject every bin with overall profit (i.e., the sum of the profits of the accommodated non-compulsory items) less than its cost. We name these algorithms First Fit with Rejections (FFR) and Best Fit with Rejections (BFR). In Theorem 4, we prove that, even adopting this improvement, the asymptotic and the absolute worst case ratios cannot be computed.

Theorem 4. *Is it impossible to compute the asymptotic and the absolute worst case ratio for algorithms* FFR *and* BFR *for both forms of the* OGBPP.

Proof. Consider instance $I(\mu, \nu)$ composed of one bin type ($|\mathcal{T}| = 1$ and $t = 1$), μ type A compulsory items, 2ν type B non-compulsory items, and 2ν type C non-compulsory items. Let $\mathcal{C}_1 = \mathcal{C}$, $\mathcal{W}_1 = \mathcal{W}$, $w_A = \mathcal{W}$, $w_B = \frac{\mathcal{W}}{2}$, $p_B = \frac{\mathcal{C}}{2} + \epsilon$, $w_C = \frac{\mathcal{W}}{2}$, and $p_C = \frac{\mathcal{C}}{2} - 2\epsilon$, with ϵ small enough.

It can be easily verified that the optimal solution consists in μ bins, each containing one type A compulsory item, and ν bins, each containing 2 type B non-compulsory items. Type C non-compulsory items are not selected because they are not profitable. Thus,

$$\mathrm{OPT}(I(\mu, \nu)) = \mu\mathcal{C} + \nu(\mathcal{C} - 2p_A) = \mu\mathcal{C} - 2\nu\epsilon \tag{6.31}$$

Consider, now, the following possible on-line sequence of items of instance $I(\mu, \nu)$:

$$\overbrace{A \ldots A}^{\mu \text{ times}} \quad \overbrace{B \quad C \quad \ldots \quad B \quad C}^{2\nu \text{ times}}$$

which, both for FFR and BFR, yields μ bins, each containing one type A compulsory item, and ν bins, each containing one type A and one type B non-compulsory items. However, bins containing non-compulsory items will be discarded because $p_B + p_C = \mathcal{C} - \epsilon < \mathcal{C}$. Therefore,

$$\mathrm{FFR}(I(\mu, \nu)) = \mathrm{BFR}(I(\mu, \nu)) = \mu\mathcal{C}. \tag{6.32}$$

The theorem holds applying Lemma 1 with $\alpha = \mathcal{C}$, $\beta = \mathcal{C}$, and $\gamma = 2\epsilon$. □

A further improvement of the algorithms presented so far consists in storing part of the items arriving on-line into a buffer before loading them somewhere. Let $k > 1$ be the size of the buffer, that is the number of items the buffer is able to store. When the buffer is full, the items are loaded into the already open bins or, if necessary, into new bins with the least ratio of cost over volume. At the end of the process, bins with an overall profit less than the cost of the bin itself will be rejected. We name First Fit with Buffer and Rejections (FFBR) this algorithm. To show the utility of using a buffer before accommodating part of the arrived items

into bins, we give an example referring to instance $I(\mu, \nu)$ of Theorem 4. Let the buffer size be 4 and let μ be a multiple of 4, and ν a multiple of 2, i.e., $\mu = 4\xi$ and $\nu = 2\upsilon$, with $\xi, \upsilon \in \mathbb{N}$. Let the items arrive on-line to a decision maker according to the following sequence:

$$\overbrace{A \ldots A}^{4\xi \text{ times}} \quad \overbrace{B \quad C \quad \ldots \quad B \quad C}^{4\upsilon \text{ times}}$$

After four type A compulsory items have arrived to a decision maker, the buffer is full and, since the received items are compulsory, they are all loaded into the bins. This process is repeated until all the type A items have arrived. Then, after accommodating all the compulsory type A items, every time the buffer is full we have two type B non-compulsory items and two type C non-compulsory items into the buffer. The decision maker will load the two type B items together into a bin and will reject the two type C items because they are not profitable. Consequently, the final solution will be the optimal solution. Nevertheless, although this improved approach seems to be very promising, in Theorem 5 we prove that even retaining some items into a buffer is not enough to guarantee asymptotic and absolute worst case ratios.

Theorem 5. *It is impossible to compute the asymptotic and the absolute worst case ratio for algorithm* FFBR *for both forms of the* OGBPP.

Proof. Given the size $k > 1$ of the buffer, consider instance $I^k(\mu, \nu)$ consisting of one bin type ($|\mathcal{T}| = 1$ and $t = 1$), $2\mu k$ type A compulsory items, $2\nu k$ type B non-compulsory items, and $2\nu k(k-1)$ type C non-compulsory items. Let $\mathcal{C}_1 = \mathcal{C}$, $\mathcal{W}_1 = \mathcal{W}$, $w_A = \mathcal{W}$, $w_B = \frac{\mathcal{W}}{k}$, $p_B = \frac{\mathcal{C}}{k} + \epsilon$, $w_C = \frac{\mathcal{W}}{k}$, and $p_C = \frac{\mathcal{C}}{k} - \frac{2\epsilon}{k-1}$, with $\epsilon > 0$ small enough, and $\mathcal{C} > 2k\epsilon$.

Note that at most k type B items can be accommodated into one bin. Also, at most k type C items can be accommodated into one bin. We have that $kp_B = \mathcal{C} + k\epsilon > \mathcal{C} = \mathcal{C}_1$ and $kp_C = \mathcal{C} - \frac{2k\epsilon}{k-1} < \mathcal{C} = \mathcal{C}_1$. Consequently, the optimum consists in $2\mu k$ bins, each containing one type A compulsory item, and 2ν bins, each containing k type B non-compulsory items:

$$\mathrm{OPT}(I^k(\mu, \nu)) = 2\mu k\mathcal{C} + 2\nu(\mathcal{C} - kp_B) = 2\mu k\mathcal{C} - 2\nu k\epsilon \tag{6.33}$$

Consider the following on-line sequence of items of instance $I^k(\mu, \nu)$ for algorithm FFBR:

$$\overbrace{\underbrace{A \ \ldots \ A}_{k \text{ times}} \ \ldots \ \underbrace{A \ \ldots \ A}_{k \text{ times}}}^{2\mu \text{ times}} \quad \overbrace{B \ \underbrace{C \ \ldots \ C}_{k-1 \text{ times}} \ \ldots \ B \ \underbrace{C \ \ldots \ C}_{k-1 \text{ times}}}^{2\nu k \text{ times}}$$

Then, applying FFBR to the above sequence, we have $2\mu k$ bins, each containing one type A compulsory item, and $2\nu k$ bins, each containing one type B non-compulsory item and $k-1$ type C non-compulsory items. However, bins containing non-compulsory items will be rejected because $p_B + (k-1)p_C = \mathcal{C} - \epsilon < \mathcal{C} = \mathcal{C}_1$. Therefore, we have:

$$\text{FFBR}(I^k(\mu, \nu)) = 2\mu k\mathcal{C} \tag{6.34}$$

and the theorem holds applying Lemma 1 with $\alpha = 2k\mathcal{C}$, $\beta = 2k\mathcal{C}$, and $\gamma = 2k\epsilon$. □

A variant of the on-line algorithm FFBR is the temporary buffer, where items are periodically loaded every time interval of length τ. We name this variant First Fit with Time Buffer and Rejections (FFTBR). FFTBR describes the case of shipping freight through means of transport with cadenced departure times. Note that, for FFTBR, the "capacity" of the buffer is not the maximum number of items k (as for FFBR) but the length of the time-slot τ between two departures. We prove in Theorem 6 that even for the on-line algorithm FFTBR it is impossible to guarantee an asymptotic and an absolute worst case ratio.

Theorem 6. *It is impossible to compute the asymptotic and the absolute worst case ratio for algorithm* FFTBR *for both the forms of the* OGBPP.

Proof. The theorem trivially holds considering instance $I^k(\mu, \nu)$ of Theorem 5 where k items fall within each time-slot of length τ. □

From Theorem 5 and Theorem 6, we can conclude that even the knowledge of part of the instance through batches (i.e., the items into the buffer) is not enough to compute the asymptotic and the absolute worst case ratios. The case limit is to have the full knowledge of the instance but, at this point, any algorithm becomes

off-line. This is the case, for instance, of the FFD and BFD heuristics already introduces in Section 3.3.1. In Theorem 7, we prove a very strong results: even for the FFD and for the BFD heuristics, which are *off-line* algorithms, it is impossible to guarantee asymptotic and absolute worst case ratios.

Theorem 7. *It is impossible to compute the asymptotic and the absolute worst case ratios for the* FFD *and the* BFD *heuristics for both forms of the* GBPP.

Proof. Consider instance $I(\mu, \nu)$ consisting of two bin types, $\mathcal{T} = \{1, 2\}$, μ type A compulsory items, 4ν type B non-compulsory items, 6ν type C non-compulsory items, and 2ν type D non-compulsory items. Let $\mathcal{W}_1 = \mathcal{W}$, $\mathcal{C}_1 = \mathcal{C}$, $\mathcal{W}_2 = \frac{72}{75}\mathcal{W}$, $\mathcal{C}_2 = \frac{49}{50}\mathcal{C}$, $w_A = \frac{72}{75}\mathcal{W}$, $w_B = \frac{18}{75}\mathcal{W}$, $p_B = \frac{22}{100}\mathcal{C}$, $w_C = \frac{25}{75}\mathcal{W}$, $p_C = \frac{32}{100}\mathcal{C}$, $w_D = \frac{39}{75}\mathcal{W}$, and $p_D = \frac{57}{100}\mathcal{C}$. It is easy to verify that the optimal solution consists in μ type 2 bins each containing one type A compulsory item, and 2ν type 1 bins each containing two type B non-compulsory items and one type D non-compulsory items:

$$\mathrm{OPT}(I(\mu, \nu)) = \mu\mathcal{C}_2 + 2\nu(\mathcal{C}_1 - 2p_B - p_D) = \frac{49}{50}\mathcal{C}\mu - \frac{\mathcal{C}}{50}\nu \tag{6.35}$$

Applying any among the four sorting rules listed in Section 3.3.1, FFD and BFD use type 1 bins and pack first all the type A compulsory items, then all the type D non-compulsory, all the type C non-compulsory items, and finally all the type B non-compulsory items. After packing all the type A compulsory items, each into type 1 bin, type D items must be accommodated. Since only one type D item can be accommodated into one type 1 bin, the PROFITABLE procedure (cf. Section 3.3.1) scans the succeeding items: those of type C. Only one type C item can be accommodated with one type D item into one type 1 bin, say b. The level of bin b is $w_C + w_D = \frac{64}{75}\mathcal{W}$ and its residual space is not enough to load any type B item. The overall profit of bin b is $P_b = p_C + p_D = \frac{89}{100}\mathcal{C} < \mathcal{C}_1$. Therefore all type D will be rejected from algorithms FFD and BFD. When scanning type C items we see that at most three of them can be accommodated into one type 1 bin, because $3w_C = \mathcal{W} = \mathcal{W}_1$ but $3p_C = \frac{96}{100}\mathcal{C} < \mathcal{C}_1$; therefore the PROFITABLE procedure rejects all the type C items in the list *SIL* with two type C succeeding items. Indeed this is not the case for the next to last and for the last type C items in the list *SIL*. More precisely, when the next to last type C compulsory item must be accommodated, the PROFITABLE procedure computes the overall profit P_b taking into account that

there are two more type C items and then type B items follow in the list. Loading two type C items into one type 1 bin, there is room only for one more type B item (because $2w_C + w_B = \frac{68}{75}\mathcal{W} < \mathcal{W}_1$, but $2w_C + 2w_B = \frac{86}{75}\mathcal{W} > \mathcal{W}_1$). The overall profit P_b is then $2p_C + p_B = \frac{86}{100}\mathcal{C} < \mathcal{C}_1$; therefore even the next to last type C bin will be discarded. Even the last type C bin will be discarded by the PROFITABLE procedure because it can be loaded with at most two type B items, but $p_C + 2p_B = \frac{76}{100}\mathcal{C} < \mathcal{C}_1$. Finally, all the type B items will be discarded because at most 4 of them can be loaded into one type 1 bin but $4p_B = \frac{88}{100}\mathcal{C} < \mathcal{C}_1$. Therefore,

$$\mathrm{FFD}(I(\mu, \nu)) = \mathrm{BFD}(I(\mu, \nu)) = \mu\mathcal{C}_1 = \mu\mathcal{C} \tag{6.36}$$

and the thesis holds applying Lemma 1 with $\alpha = \mathcal{C}$, $\beta = \frac{49}{50}\mathcal{C}$, and $\gamma = \frac{\mathcal{C}}{50}$. □

6.4 The On-line Generalized Bin Packing Problem with item profits proportional to item volumes

In this section, we analyze the algorithms studied in section 6.3 within the particular setting of the OGBPP_κ, where item profits are proportional to item volumes through a positive coefficient κ. As mentioned in Chapter 1 and in Section 6.2.3, this is an important setting because in several applications shipping costs are proportional to item volumes. We prove that it is impossible to conduct a worst-case analysis for all the algorithms already presented in Section 6.3, even in the simpler setting of the OGBPP_κ, for both forms of the problem. We start with Theorem 8, where we prove that it is impossible to compute the asymptotic and absolute worst case ratios for algorithms FF and BF when applied to the OGBPP_κ.

Theorem 8. *It is impossible to compute the asymptotic and the absolute worst case ratios for algorithms* FF *and* BF *for both forms of the* OGBPP_κ.

Proof. Consider instance $I(\nu_A, \nu_B, \nu_C)$, made up by one bin type with volume $\mathcal{W}$ and cost $\mathcal{C}$, ν_A type A compulsory items, ν_B type B non-compulsory items, and ν_C type C non-compulsory items. Let $w_A = \left(\frac{1}{2} + \epsilon\right)\mathcal{W}$, $w_B = \left(\frac{1}{2} + \epsilon\right)\mathcal{W}$, and $w_C = 1 - \epsilon$, $\kappa = (1 + 2\epsilon)\frac{\mathcal{C}}{\mathcal{W}}$, and $0 < \epsilon < \frac{-1+\sqrt{2}}{2}$.

We have $\kappa\, w_B = (1+2\epsilon)\left(\frac{1}{2}+\epsilon\right)\mathcal{C} = \frac{4\epsilon^2+4\epsilon+1}{2}\mathcal{C}$. Note that, by construction, any item cannot be loaded with any other item, that is a bin can accommodate one item only. In order to let type B items be profitable, one should have $\kappa\, w_B > \mathcal{C}$, which, after some manipulations, would imply $\epsilon < \frac{-1-\sqrt{2}}{2}$ or $\epsilon > \frac{-1+\sqrt{2}}{2}$, both contradicting the requirement that $0 < \epsilon < \frac{-1+\sqrt{2}}{2}$. Therefore type B items are not profitable and, since none of them can be loaded with any other item, they will not appear in the optimal solution. Concerning type C items, we have $\kappa w_C = (1+2\epsilon)(1-\epsilon)\mathcal{C} = (1+\epsilon-2\epsilon^2)\,\mathcal{C}$. Imposing $\kappa w_C > \mathcal{C}$, we get $\epsilon(1-2\epsilon) > 0$, which implies $0 < \epsilon < \frac{1}{2}$. Thence, since $\frac{-1+\sqrt{2}}{2} < \frac{1}{2}$, we have that, for any choice of ϵ according to requirement on ϵ, type C items are always profitable.

Therefore, the optimal solution consists in ν_A bins, each containing one type A compulsory item and ν_C bins, each containing one tpye C non-compulsory item:

$$\mathrm{OPT}(I(\nu_A,\, \nu_B,\, \nu_C)) = \nu_A\mathcal{C} + \nu_C(\mathcal{C} - \kappa w_C) = \nu_A\mathcal{C} + \nu_C\left(2\epsilon^2 - \epsilon\right)\mathcal{C} \tag{6.37}$$

Note that $2\epsilon^2 - \epsilon$ is negative for $0 < \epsilon < \frac{-1+\sqrt{2}}{2}$. For any on-line sequence of items, FF selects ν_A bins, each containing one type A compulsory item, ν_B bins, each containing one type B non-compulsory item, and ν_C bins, each containing one type C non-compulsory item

$$\begin{aligned}
\mathrm{FF}(I(\nu_A,\, \nu_B,\, \nu_C)) &= \mathrm{BF}(I(\nu_A,\, \nu_B,\, \nu_C)) = \\
&= \nu_A\mathcal{C} + \nu_B(\mathcal{C} - \kappa w_B) + \nu_C(\mathcal{C} - \kappa w_C) = \\
&= \nu_A\mathcal{C} + \nu_B\frac{1-4\epsilon-4\epsilon^2}{2}\mathcal{C} + \nu_C\left(2\epsilon^2 - \epsilon\right)\mathcal{C}
\end{aligned} \tag{6.38}$$

Note that $1-4\epsilon-4\epsilon^2$ is positive for $0 < \epsilon < \frac{-1+\sqrt{2}}{2}$.

According to (6.1), in order to compute the asymptotic worst case ratio, we have to find a proper constant $\mathcal{O}(1)$ and the least $\mathcal{R}$, both independent of the instances, such that

$$\nu_A\mathcal{C} + \nu_B\frac{1-4\epsilon-4\epsilon^2}{2}\mathcal{C} + \nu_C\left(2\epsilon^2 - \epsilon\right)\mathcal{C} \leq \mathcal{R}\left(\nu_A\mathcal{C} + \nu_C\left(2\epsilon^2 - \epsilon\right)\mathcal{C}\right) + \mathcal{O}(1) \tag{6.39}$$

If we consider instance $I(0,\, \nu_B,\, 0)$, (6.39) becomes

$$\nu_B \frac{1-4\epsilon-4\epsilon^2}{2}\mathcal{C} \leq \mathcal{O}(1), \tag{6.40}$$

which is impossible because a constant cannot be greater than a linear positive ($\mathcal{O}(\nu)$) term.

According to (6.2), in order to compute the absolute worst case ratio, we have to find the least ρ, independent of the instances, such that

$$\nu_A\mathcal{C} + \nu_B \frac{1-4\epsilon-4\epsilon^2}{2}\mathcal{C} + \nu_C\left(2\epsilon^2-\epsilon\right)\mathcal{C} \leq \rho\left(\nu_A\mathcal{C} + \nu_C\left(2\epsilon^2-\epsilon\right)\mathcal{C}\right) \tag{6.41}$$

If we consider instance $I(1,\, \nu_B,\, 0)$, (6.41) becomes

$$\mathcal{C} + \nu_B \frac{1-4\epsilon-4\epsilon^2}{2}\mathcal{C} \leq \rho\mathcal{C}, \tag{6.42}$$

which implies $\rho \to +\infty$, since ν_B can be arbitrarily large. On the contrary, if we consider instance $I(0,\, \nu_B,\, 1)$, (6.41) becomes

$$\nu_B \frac{1-4\epsilon-4\epsilon^2}{2}\mathcal{C} + \left(2\epsilon^2-\epsilon\right)\mathcal{C} \leq \rho\left(\left(2\epsilon^2-\epsilon\right)\mathcal{C}\right), \tag{6.43}$$

which, since $2\epsilon^2-\epsilon < 0$ and since ν_B can be arbitrarily large, implies $\rho \to -\infty$. But this contradicts $\rho \to +\infty$ in order to satisfy (6.42). Therefore it is impossible even to compute the absolute worst case ratio.

Considering the maximization form of the problem we have

$$\mathrm{OPT}(I(\nu_A,\, \nu_B,\, \nu_C)) = -\nu_A\mathcal{C} - \nu_C(\mathcal{C}-\kappa w_C) = -\nu_A\mathcal{C} - \nu_C\left(2\epsilon^2-\epsilon\right)\mathcal{C} \tag{6.44}$$

and

$$\begin{aligned} \mathrm{FF}(I(\nu_A,\, \nu_B,\, \nu_C)) &= \mathrm{BF}(I(\nu_A,\, \nu_B,\, \nu_C)) = \\ &= -\nu_A\mathcal{C} - \nu_B(\mathcal{C}-\kappa w_B) - \nu_C(\mathcal{C}-\kappa w_C) = \\ &= -\nu_A\mathcal{C} - \nu_B\frac{1-4\epsilon-4\epsilon^2}{2}\mathcal{C} - \nu_C\left(2\epsilon^2-\epsilon\right)\mathcal{C} \end{aligned} \tag{6.45}$$

According to (6.5), in order to compute the asymptotic worst case ratio, we have to find a proper constant $\mathcal{O}(1)$ and the greatest $\mathcal{R}$, both independent of the instances, such that

$$-\nu_A\mathcal{C}-\nu_B\frac{1-4\epsilon-4\epsilon^2}{2}\mathcal{C}-\nu_C\left(2\epsilon^2-\epsilon\right)\mathcal{C}\geq\mathcal{R}\left(-\nu_A\mathcal{C}-\nu_C\left(2\epsilon^2-\epsilon\right)\mathcal{C}\right)+\mathcal{O}(1) \quad (6.46)$$

If we consider instance $I(0,\,\nu_B,\,0)$, (6.46) becomes

$$-\nu_B\frac{1-4\epsilon-4\epsilon^2}{2}\mathcal{C}\geq\mathcal{O}(1), \quad (6.47)$$

which is impossible because a constant cannot be less than a negative linear (- $\mathcal{O}(\nu)$) term.

According to (6.6), in order to compute the absolute worst case ratio, we have to find the greatest ρ, independent of the instances, such that

$$-\nu_A\mathcal{C}-\nu_B\frac{1-4\epsilon-4\epsilon^2}{2}\mathcal{C}-\nu_C\left(2\epsilon^2-\epsilon\right)\mathcal{C}\geq\rho\left(-\nu_A\mathcal{C}-\nu_C\left(2\epsilon^2-\epsilon\right)\mathcal{C}\right) \quad (6.48)$$

Considering instance $I(1,\,\nu_B,\,0)$, (6.48) becomes

$$-\mathcal{C}-\nu_B\frac{1-4\epsilon-4\epsilon^2}{2}\mathcal{C}\geq-\rho\mathcal{C}, \quad (6.49)$$

which implies $\rho\to+\infty$, since ν_B can be arbitrarily large. On the contrary, if we consider instance $I(0,\,\nu_B,\,1)$, (6.48) becomes

$$-\nu_B\frac{1-4\epsilon-4\epsilon^2}{2}\mathcal{C}-\left(2\epsilon^2-\epsilon\right)\mathcal{C}\geq\rho\left(2\epsilon^2-\epsilon\right)\mathcal{C}, \quad (6.50)$$

which, since $2\epsilon^2-\epsilon<0$ and since ν_B can be arbitrarily large, implies $\rho\to-\infty$. But this contradicts $\rho\to+\infty$ in order to satisfy (6.49). □

In Theorem 9, we prove that it is impossible to conduct a worst case analysis even for algorithms FFR and BFR.

Theorem 9. *It is impossible to compute the asymptotic and the absolute worst case ratios for* FFR *and* BFR *for both forms of the* OGBPP$_\kappa$.

Proof. Consider instance $I(\mu, \nu)$ composed of one bin type with volume $\mathcal{W}$ and cost $\mathcal{C}$, μ type A compulsory items, 2ν type B non-compulsory items, and 2ν type C non-compulsory items. Let $w_A = \mathcal{W}$, $w_B = \left(\frac{1}{2} - \epsilon\right)\mathcal{W}$, $w_C = \left(\frac{1}{2} + \epsilon\right)\mathcal{W}$, and $\kappa = (1 + 2\epsilon)\frac{\mathcal{C}}{\mathcal{W}}$, with $0 < \epsilon < \frac{-1+\sqrt{2}}{2}$.

It is easy to verify that the optimal solution consists in μ bins each containing one type A compulsory item and 2ν bins each containing one type B and one type C non-compulsory item. Indeed we have

$$\kappa(w_B + w_C) = (1 + 2\epsilon)\mathcal{C} > \mathcal{C} \tag{6.51}$$

Thence

$$\mathrm{OPT}(I(\mu, \nu)) = \mu\mathcal{C} + 2\nu(\mathcal{C} - \kappa(w_B + w_C)) = \mu\mathcal{C} - 4\epsilon\nu\mathcal{C} \tag{6.52}$$

The sequence

$$\overbrace{A \quad \ldots \quad A}^{\mu \text{ times}} \quad \overbrace{B \quad \ldots \quad B}^{2\nu \text{ times}} \quad \overbrace{C \quad \ldots \quad C}^{2\nu \text{ times}}$$

yields μ bins each containing one type A compulsory items, ν bins each containing two type B non-compulsory items, and 2ν bins each containing one type C non-compulsory item. Bins containing two type B non-compulsory are not profitable because

$$2\kappa w_B = \left(1 - 4\epsilon^2\right)\mathcal{C} < \mathcal{C} \tag{6.53}$$

Similarly, bins containing one type C non-compulsory item are not profitable because

$$\kappa w_C = \frac{(1 + 2\epsilon)^2}{2}\mathcal{C} < \frac{\left(1 + 2\frac{-1+\sqrt{2}}{2}\right)^2}{2}\mathcal{C} < \mathcal{C} \tag{6.54}$$

Therefore only bins containing one type A compulsory item will be retained by both FFR and BFR, and we have:

$$\mathrm{FFR}(I(\mu, \nu)) = \mathrm{BFR}(I(\mu, \nu)) = \mu\mathcal{C} \tag{6.55}$$

The theorem holds applying Lemma 1 with $\alpha = \mathcal{C}$, $\beta = \mathcal{C}$, and $\gamma = 4\epsilon\mathcal{C}$. □

In Theorem 10, we prove that it is impossible to compute the asymptotic and worst case ratios even for algorithm FFBR.

Theorem 10. *It is impossible to compute the asymptotic and absolute worst case ratios for* FFBR *for both forms of the* OGBPP_κ.

Proof. Given the size $k > 1$ of the buffer, consider instance $I^k(\mu, \nu)$ composed of one bin type with volume $\mathcal{W}$ and cost $\mathcal{C}$, μk type A compulsory items, 2ν type B non-compulsory items, and $2\nu(k-1)$ type C non-compulsory items. Let $w_A = \mathcal{W}$, $w_B = \frac{W}{2}$, $w_C = \frac{1-2\epsilon}{k-1}\frac{W}{2}$, and $\kappa = (1+\epsilon)\frac{\mathcal{C}}{\mathcal{W}}$, with $\epsilon > 0$ small enough. It is easy to verify that the optimal solution consists in μk bins each containing one type A compulsory item, and ν bins each containing two type B non-compulsory items:

$$\mathrm{OPT}(I^k(\mu, \nu)) = \mu k\mathcal{C} + \nu(\mathcal{C} - 2\kappa w_B) = \mu k\mathcal{C} - \nu\epsilon\mathcal{C} \tag{6.56}$$

The sequence

$$\overbrace{\underbrace{A \ \ldots \ A}_{k \text{ times}} \ \ldots \ \underbrace{A \ \ldots \ A}_{k \text{ times}}}^{\mu \text{ times}} \quad \overbrace{B \ \underbrace{C \ \ldots \ C}_{k-1 \text{ times}} \ \ldots \ B \ \underbrace{C \ \ldots \ C}_{k-1 \text{ times}}}^{2\nu \text{ times}}$$

yields μk bins each containing one type A compulsory items and 2ν bins each containing one type B and $k-1$ type C non-compulsory items. The profit of these bins is

$$\kappa(w_B + (k-1)w_C) = \left(1 - \epsilon^2\right)\mathcal{C} < \mathcal{C} \tag{6.57}$$

Therefore only bins containing compulsory items are retained, and we have

$$\mathrm{FFBR}(I^k(\mu, \nu)) = \mu k\mathcal{C} \tag{6.58}$$

The theorem holds applying Lemma 1 with $\alpha = k\mathcal{C}$, $\beta = k\mathcal{C}$, and $\gamma = \epsilon\mathcal{C}$. □

In Theorem 11, we show that it is impossible to compute the asymptotic and absolute worst case ratios even for algorithm FFTBR.

Theorem 11. *It is impossible to compute the asymptotic and absolute worst case ratios for* FFTBR *for both the forms of the* OGBPP_κ.

Proof. The theorem trivially holds considering instance $I^k(\mu, \nu)$ of Theorem 10 where k items fall within each time-slot of length τ. □

We conclude this section showing that even for the off-line algorithms FFD and BFD presented in Section 3.3.1 it is impossible to guarantee asymptotic and absolute worst case ratio when applied to the GBPP_κ.

Theorem 12. *It is impossible to compute the asymptotic and absolute worst case ratios for the* FFD *and* BFD *algorithms for both forms of the* GBPP_κ.

Proof. Consider instance $I(\mu, \nu)$ composed of one bin type with capacity $\mathcal{W}$ and cost $\mathcal{C}$, μ type A compulsory items, 4ν type B non-compulsory items, ν type C non-compulsory items, and ν type D non-compulsory items. Let $w_A = \mathcal{W}$, $w_B = \left(\frac{1}{4} - \epsilon\right)\mathcal{W}$, $w_C = \left(\frac{1}{2} - 3\epsilon\right)\mathcal{W}$, $w_D = \left(\frac{1}{2} + 2\epsilon\right)\mathcal{W}$, with $\kappa = (1+\epsilon)\frac{\mathcal{C}}{\mathcal{W}}$, and $0 < \epsilon < \frac{1}{28}$. The optimal solution consists in μ bins each containing one type A compulsory item and ν bins each containing two type B non-compulsory items and one type D non-compulsory item. Indeed we have

$$\kappa(2w_B + w_D) = (1 + \epsilon)\mathcal{C} > \mathcal{C}. \tag{6.59}$$

To prove that this combination is optimal, we show that other profitable combinations do not exist. Combination (C, D) is not profitable because

$$\kappa(w_C + w_D) = \left(1 - \epsilon^2\right)\mathcal{C} < \mathcal{C} \tag{6.60}$$

Moreover, it is not possible to fill the residual space (equal to ϵ) with one type B item. To do that, we should have $w_B \leq \epsilon$, which would imply $\epsilon \geq \frac{1}{8}$; but this is impossible because, by construction, $\epsilon < \frac{1}{28}$. Combination (D, D) is clearly not possible because $2w_D = (1 + 4\epsilon)\mathcal{W} > \mathcal{W}$. Combination (C, C, C) is not possible because $3w_C = \frac{3}{2} - 9\epsilon$. It would be possible if we had $3w_C \leq \mathcal{W}$, which implies $\epsilon \geq \frac{1}{18}$, contradicting $\epsilon < \frac{1}{28}$. Combination (C, C, B) is not possible because, imposing $2w_C + w_B \leq \mathcal{W}$ we get $\epsilon \geq \frac{1}{28}$, contradicting the requirement $\epsilon < \frac{1}{28}$. Combination (B, B, C) is not profitable because

$$\kappa(2w_B + w_C) = (1 + \epsilon)(1 - 5\epsilon)\mathcal{C} < (1 + \epsilon)(1 - \epsilon)\mathcal{C} = \left(1 - \epsilon^2\right)\mathcal{C} < \mathcal{C} \tag{6.61}$$

Finally, we can load at most 4 type B items into a bin. To be able to load more we should have $5w_B \leq \mathcal{W}$, which would imply $\epsilon \geq \frac{1}{20}$, contradicting the requirement $\epsilon \geq \frac{1}{28}$. However

$$4\kappa w_B = \left(1 - \epsilon^2\right)\mathcal{C} < \mathcal{C}, \tag{6.62}$$

which means that even combination (B, B, B, B) is not profitable. Therefore, the only profitable combination is (B, B, D) and the optimum is

$$\mathrm{OPT}(I(\mu, \nu)) = \mu\mathcal{C} + \nu(\mathcal{C} - \kappa(2w_B + w_D) = \mu\mathcal{C} - \nu\epsilon\mathcal{C}. \tag{6.63}$$

Applying any of the four sorting criteria shown in Section 6.3, we get the following ordering of items:

$$\overbrace{A \quad \ldots \quad A}^{\mu \text{ times}} \quad \overbrace{D \quad \ldots \quad D}^{\nu \text{ times}} \quad \overbrace{C \quad \ldots \quad C}^{\nu \text{ times}} \quad \overbrace{B \quad \ldots \quad B}^{4\nu \text{ times}}$$

After placing all the type A compulsory items, the PROFITABLE function (cf. Section 3.3.1) will scan all the non-compulsory items in the sequence above. One type D item will be associated to one type C item because type C items are between type D and type B items. Consequently combination (D, B, B) will never be encountered when performing all the PROFITABLE functions. Since all the other combinations of items are not profitable, then all the non-compulsory items will be rejected. We then have

$$\mathrm{FFR}(I(\mu, \nu)) = \mathrm{BFR}(I(\mu, \nu)) = \mu\mathcal{C}. \tag{6.64}$$

The theorem holds applying Lemma 1 with $\alpha = \mathcal{C}$, $\beta = \mathcal{C}$, and $\gamma = \epsilon\mathcal{C}$. □

6.5 First Fit algorithm for the On-line Variable Cost and Size Bin Packing Problem

In this section, we study the FF algorithm in the particular setting of the OVCS-BPP. As mentioned in Section 6.3, FF works as follows. Every time an item $i \in \mathcal{I}$ arrives on-line, we try to load it into the first bin (among the already open ones)

able to contain it. If none of the open bins is able to accommodate item $i \in \mathcal{I}$, we select a new bin in order to load the received item. In the OVCSBPP, we choose the new bin to be of type s i.e., as stated in Section 6.2.5, with the least cost over volume ratio. Recall from Section 2 that Johnson [1974] proved that the asymptotic worst case ratio for FF applied to the BPP is 17/10 and Li and Chen [2006] proved that this ratio worsens to 2 when FF is applied to a variant of the BPP named Bin Packing Problem with concave costs of bin utilization. Although the OVCSBPP is a generalization of the problem studied by Li and Chen [2006], in Theorem 13 we prove that we can still guarantee an asymptotic worst case ratio equal to 2 for algorithm FF. Moreover, we prove that this bound is tight.

Theorem 13. *The asymptotic worst case ratio of algorithm* FF *applied to the* OVCSBPP *is 2 and this bound is tight.*

Proof. Given two consecutive bins j and $j+1$ in the solution yielded by FF for instance I, the sum of the levels of these bins must be greater than $\mathcal{W}_s$, otherwise the items in bin $j+1$ could have been accommodated into bin j. Formally:

$$\beta(j) + \beta(j+1) > \mathcal{W}_s. \tag{6.65}$$

Summing side by side (6.65) with j ranging from 1 to $f-1$ we get

$$\sum_{j=1}^{f-1} \beta(j) + \sum_{j=1}^{f-1} \beta(j+1) > \sum_{j=1}^{f-1} \mathcal{W}_s. \tag{6.66}$$

Note that j ranges up to $f-1$ and not up to f otherwise we would get $\beta(f+1)$ for $j = f$ in the second summation on the left side of inequality (6.66), which does not exist. Changing the range within the second summation on the left side of the inequality and computing the summation on the right side of inequality (6.66) we have

$$\sum_{j=1}^{f-1} \beta(j) + \sum_{j=2}^{f} \beta(j) > (f-1)\mathcal{W}_s. \tag{6.67}$$

The two summations on the left side of inequality (6.67) are very similar. In order to gather them into a unique summation of the form $\sum_{j=1}^{f} \beta(j)$, we add and subtract both $\beta(1)$ and $\beta(f)$ on the first side of inequality (6.67) as follows

$$\sum_{j=1}^{f-1} \beta(j) + \beta(f) - \beta(f) + \sum_{j=2}^{f} \beta(j) + \beta(1) - \beta(1) > (f-1)\mathcal{W}_s, \tag{6.68}$$

$$\sum_{j=1}^{f} \beta(j) - \beta(f) + \sum_{j=1}^{f} \beta(j) - \beta(1) > (f-1)\mathcal{W}_s. \tag{6.69}$$

It is now possible to collect the two summations into a unique one and develop the following transformations:

$$2\sum_{j=1}^{f} \beta(j) - \beta(1) - \beta(f) > (f-1)\mathcal{W}_s, \tag{6.70}$$

$$2\sum_{j=1}^{f} \beta(j) > (f-1)\mathcal{W}_s + \beta(1) + \beta(f) > (f-1)\mathcal{W}_s. \tag{6.71}$$

Therefore,

$$(f-1)\mathcal{W}_s < 2\sum_{j=1}^{f} \beta(j) = 2\sum_{j\in B} \beta(j). \tag{6.72}$$

Since FF uses f bins of type s, then the cost of its solution is $f \cdot \mathcal{C}_s$ and we have

$$\mathrm{FF}(I) = f \cdot \mathcal{C}_s = (f-1+1)\mathcal{C}_s = (f-1)\mathcal{C}_s + \mathcal{C}_s = (f-1)\mathcal{W}_s\frac{\mathcal{C}_s}{\mathcal{W}_s} + \mathcal{C}_s. \tag{6.73}$$

Plugging (6.72) into (6.73) we get

$$\mathrm{FF}(I) < 2\frac{\mathcal{C}_s}{\mathcal{W}_s}\sum_{j=1}^{f} \beta(j) + \mathcal{C}_s. \tag{6.74}$$

Since all the items are taken, the sum of the levels of the bins employed by the FF algorithm is equal to the sum of the levels of the bins (the set B^*) of an optimal solution. We then have

$$\mathrm{FF}(I) < 2\frac{\mathcal{C}_s}{\mathcal{W}_s}\sum_{j=1}^{f} \beta(j) + \mathcal{C}_s = 2\frac{\mathcal{C}_s}{\mathcal{W}_s}\sum_{j=1}^{g} \beta(j) + \mathcal{C}_s. \tag{6.75}$$

Since FF uses bins of type s, we have that

$$\frac{\mathcal{C}_s}{\mathcal{W}_s} \leq \frac{\mathcal{C}_t}{\mathcal{W}_t}, \qquad \forall t \in \mathcal{T}. \tag{6.76}$$

Considering the indicator function $\sigma(j)$ introduced in Section 3.1.1 and the set of bins B^* of an optimal solution, (6.76) becomes:

$$\frac{\mathcal{C}_s}{\mathcal{W}_s} \leq \frac{\mathcal{C}_{\sigma(j)}}{\mathcal{W}_{\sigma(j)}}, \qquad \forall j \in B^*. \tag{6.77}$$

By means of (6.77), inequality (6.75) can be transformed as follows

$$\mathrm{FF}(I) < 2\frac{\mathcal{C}_s}{\mathcal{W}_s}\sum_{j=1}^{g}\beta(j) + \mathcal{C}_s = 2\sum_{j=1}^{g}\frac{\mathcal{C}_s}{\mathcal{W}_s}\beta(j) + \mathcal{C}_s \leq 2\sum_{j=1}^{g}\frac{\mathcal{C}_{\sigma(j)}}{\mathcal{W}_{\sigma(j)}}\beta(j) + \mathcal{C}_s. \tag{6.78}$$

The level of a bin must be less or equal the capacity of the bin itself, i.e., $\beta(j) \leq \mathcal{W}_{\sigma(j)}$. Plugging this inequality into (6.78), we get

$$\mathrm{FF}(I) < 2\sum_{j=1}^{g}\frac{\mathcal{C}_{\sigma(j)}}{\mathcal{W}_{\sigma(j)}}W_{\sigma(j)} + \mathcal{C}_s = 2\sum_{j=1}^{g}C_{\sigma(j)} + \mathcal{C}_s. \tag{6.79}$$

But $\sum_{j=1}^{g} C_{\sigma(j)}$ is the cost of an optimal solution, therefore (6.79) becomes

$$\mathrm{FF}(I) < 2 \cdot \mathrm{OPT}(I) + \ C_s, \tag{6.80}$$

which means that algorithm FF has an asymptotic worst-case ratio of 2.

To prove that the bound is tight consider instance I composed of n items with volume $w = \frac{1}{2} + \epsilon$ and two bin types, 1 and 2, with $\mathcal{C}_1 = 1$, $\mathcal{W}_1 = 1 + \epsilon$, $\mathcal{C}_2 = \frac{1}{2} + \epsilon$, and $\mathcal{W}_2 = \frac{1}{2} + \epsilon$. Note that

$$\frac{\mathcal{C}_1}{\mathcal{W}_1} = \frac{1}{1+\epsilon} \leq 1 = \frac{\mathcal{C}_2}{\mathcal{W}_2}. \tag{6.81}$$

Therefore, FF selects bins of type 1. Nevertheless, these bins are not big enough to accommodate two items because $2w = 1 + 2\epsilon > 1 + \epsilon = \mathcal{W}_1$. Thence we have

$$\mathrm{FF}(I) = n\,\mathcal{C}_1 = n. \tag{6.82}$$

On the contrary, the optimal solution consists in accommodating the n items each into a type 2 bin because, for ϵ small enough, $\mathcal{C}_2 < \mathcal{C}_1$. Then

$$\mathrm{OPT}(I) = n\,\mathcal{C}_2 = n\left(\frac{1}{2} + \epsilon\right). \tag{6.83}$$

The worst case ratio is then

$$\mathcal{R}(I) = \frac{\mathrm{FF}(I)}{\mathrm{OPT}(I)} = \frac{1}{\frac{1}{2} + \epsilon} = \frac{2}{1 + 2\epsilon}, \tag{6.84}$$

which approaches 2 as $\epsilon \to 0$. □

Chapter 7

Conclusions

In this thesis, we have introduced and studied a new family of packing problems named Generalized Bin Packing Problems. These problems are the Generalized Bin Packing Problem (GBPP), the Stochastic Generalized Bin Packing Problem (S-GBPP), the On-line Generalized Bin Packing Problem (OGBPP), the Generalized Bin Packing Problem with item profits proportional to item volumes (GBPP_κ), the On-line Generalized Bin Packing Problem with item profits proportional to item volumes (OGBPP_κ), and the On-line Variable Cost and Size Bin Packing Problem (OVCSBPP). We have addressed these new problems in order to overcome a noteworthy portion of a gap in the packing literature in terms of comprehensive study concerning the joint presence of both compulsory and non-compulsory items, and in terms of unified methodologies in order to solve packing problems with different objective functions. Moreover, we could address new real-life applications not yet addressed or only partially addressed by the current state-of-the-art packing problems. For this reason, we have both studied deterministic and stochastic problems, with two kinds of stochasticity concerning the items: 1) stochasticity of the item attributes, where one attribute is affected by uncertainty and modeled as a random variable, and 2) stochasticity of the item availability, i.e., the items are not known a priori but arrive on-line in an unpredictable way to a decision maker.

Our main result concerned the development of models and unified methodologies of these new packing problems characterized by (with the only exception of the OVCSBPP) the joint presence of both compulsory and non-compulsory items. This

innovative feature allowed us to address and collect several bin packing and knapsack problems at the same time into a unique one: the GBPP. These problems are the Bin Packing Problem (BPP), the Variable Sized Bin Packing Problem (VSBPP), the Variable Cost and Size Bin Packing Problem (VCSBPP), the Knapsack Problem (KP), the Multiple Knapsack Problem (MKP), and the Multiple Knapsack Problem with identical capacities (MKPI). Moreover, the GBPP allowed us to address new applications and to bring new contributions in logistics, in transportation, and in the waste collection problem. For this problem, we gave two formulations and we proposed variegated methods in terms of quality and computational time. We created new instance classes in order to test all these methods and we achieved excellent results, with an overall percentage mean gap of 0.03%. Moreover, we could close most of the instances in the GBPP literature.

The S-GBPP allowed us to address new applications where each profit depends on the bin where it is loaded and it is described by a random variable. These applications arise in particular in logistics, where the freight consolidation is essential to optimize the delivery process, and in the Railway Track Maintenance Planning Problem, where maintenance operations must be scheduled into time-slots and their costs are uncertain and depend on the time-slots where they are assigned. For this problem, we provided a stochastic model and, applying the extreme value theory, we derived a deterministic approximation.

Finally, we studied the on-line variant of the GBPP, the OGBPP. This problem arises in all applications where orders arrive on-line, in particular in logistics, with freight forwarders. For this problem, we studied a wide range of algorithms in order to test whether the available tools in the literature (i.e., the asymptotic and absolute worst case ratios) are still effective when a richer setting as the OGBPP is tackled. Our study revealed a strong result: we proved that, for all the proposed algorithms, it is even impossible to apply the definition of these worst case ratios. This behavior occurred also in the OGBPP_{κ}, a particular case of the OGBPP where item profits are proportional to their corresponding volumes through a positive coefficient.

We believe that the ultimate packing problem for which it is possible to compute a performance ratio is the OVCSBPP, the closest problem to the GBPP, where the items arrive on-line, but still without the presence of non-compulsory ones. For this problem, we could generalize the work of Li and Chen [2006] to a more general

setting, still guaranteeing the same performance ratio.

The study on the OGBPP is still an open problem. A very challenging development is to give a formal answer to the question whether there exists at least one on-line algorithm with any worst case ratio. This study will be the topic of future research.

Bibliography

A. Akkas. *Transportation Resource Scheduling in Food Retail Industry*. PhD thesis, Massachussets Institute of Technology, 2004.

C. Alves and J. M. Valério de Carvalho. Accelerating column generation for variable sized bin-packing problems. *European Journal of Operational Research*, 183:1333–1352, 2007.

C. Alves and J. M. Valério de Carvalho. A stabilized branch-and-price-and-cut algorithm for the multiple length cutting stock problem. *Computers & Operations Research*, 35:1315–1328, 2008.

A. Atamturk and M. W. P. Savelsberg. Integer programming software systems. *Annals of Operations Research*, 140:67–124, 2005.

M. M. Baldi, T. G. Crainic, G. Perboli, and R. Tadei. The generalized bin packing problem. *Transportation Research Part E*, 48(6):1205–1220, 2012a. doi: 10.1016/j.tre.2012.06.005.

M. M. Baldi, T. G. Crainic, G. Perboli, and R. Tadei. Branch-and-price and beam search algorithms for the variable cost and size bin packing problem with optional items. *Annals of Operations Research*, 2012b. doi: 10.1007/s10479-012-1283-2.

M. M. Baldi, G. Perboli, and R. Tadei. The three-dimensional knapsack problem with balancing constraints. *Applied Mathematics and Computation*, 218(19):9802–9818, 2012c.

C. Barnhart, E. L. Johnson, G. L. Nemhauser, M. W. P. Savelsberg, and P. H. Vance. Branch-and-price: Column generation for solving huge integer programs. *Operations Research*, 46(3):316–329, 1998.

G. Belov and G. Scheithauer. A cutting plane algorithm for the one-dimensional cutting stock problem with multiple stock lengths. *European Journal of Operational Research*, 141:274–294, 2002.

A. Bettinelli, A. Ceselli, and G. Righini. A branch-and-price algorithm for the variable size bin packing problem with minimum filling constraint. *Annals of Operations Research*, 179:221–241, 2010.

M. T. Bianchi de Aguiar. *Optimization Techniques for the Mixed Urban Rural Solid Waste Collection Problem*. PhD thesis, Faculdade de engenharia da Universidade do Porto, 2010.

M. T. Bianchi de Aguiar, M. A. Caravilla, and J. F. Oliveira. Municipal waste collection in ponte de lima, portugal - a vehicle routing application. *OR Insight*, 25:185–198, 2012.

C. Boutevin, M. Gourgand, and S. Norre. Bin packing extensions for solving an industrial line balancing problem. In *Assembly and Task Planning, 2003. Proceedings of the IEEE International Symposium on*, pages 115–121, 2003.

C. Chu and R. La. Variable-sized bin packing: Tight absolute worst-case performance ratios for four approximation algorithms. *SIAM Journal on Computing*, 30:2069–2083, 2001.

J. K. Cochran and B. Ramanujam. Carrier-mode logistics optimization of inbound supply chains for electronics manufacturing. *International Journal of Production Economics*, 103(2):826–840, 2006.

E.G. Coffman Jr., K. So, M. Hofri, and A.C. Yao. A stochastic model of bin-packing. *Information and Control*, 44(2):105–115, 1980.

A. M. Cohn and C. Barnhart. The stochastic knapsack problem with random weights: A heuristic approach to robust transportation planning. In *Proceedings of the Triennial Symposium on Transportation Analysis*, 1998.

I. Correia, L. Gouveia, and F. Saldanha-da-Gama. Solving the variable size bin packing problem with discretized formulations. *Computers & Operations Research*, 35: 2103–2113, 2008.

T. G. Crainic, G. Perboli, W. Rei, and R. Tadei. Efficient lower bounds and heuristics for the variable cost and size bin packing problem. *Computers & Operations Research*, 38:1474–1482, 2011.

T.G. Crainic, G. Perboli, M. Pezzuto, and R. Tadei. New Bin Packing Fast Lower Bounds. *Computers & Operations Research*, 34(11):3439–3457, 2007.

W.F. de la Vega and G.S. Lueker. Bin packing can be solved within $1 + \epsilon$ in linear time. *Combinatorica*, 1(4):349–355, 1981.

B. Dean, M. Goemans, and J. Vondrak. Approximating the stochastic knapsack problem: The benefit of adaptivity. *Mathematics of Operations Research*, 33: 945–964, 2008.

F. Della Croce, M. Ghirardi, and R. Tadei. Recovering beam search: Enhancing the beam search approach for combinatorial optimization problems. *Journal of Heuristics*, 10:89–104, 2004.

P. Detti, C. Hurkens, A. Agnetis, and G. Ciaschetti. Optimal packet-to-slot assignment in mobile telecommunications. *Operations Research Letters*, 37(4):261–264, 2009.

L. Epstein. On online bin packing with lib constraints. *Naval Research Logistics*, 56:780 – 786, 2009.

L. Epstein and A. Levin. An aptas for generalized cost variable-sized bin packing. *SIAM Journal on Computing*, 38(1):411–428, 2008.

L. Epstein and A. Levin. Bin packing with general cost structures. *Mathematical Programming*, 132:355 – 391, 2012.

L. Epstein and R. van Stee. Online bin packing with resource augmentation. In *Proceedings of the Second international conference on Approximation and Online Algorithms*, pages 23–35. Springer-Verlag, 2005. ISBN 978-3-540-24574-2.

L. Epstein, L. M. Favrholdt, and A. Levin. Online variable-sized bin packing with conflicts. *Discrete Optimization*, 8(2):333 – 343, 2011.

S. Fazi, T. van Woensel, and J. C. Fransoo. A stochastic variable size bin packing problem with time constraints. Technical report, Technische Universiteit Eindhoven, 2012.

S. P. Fekete and J. Schepers. New classes of lower bounds for bin packing problems. *Mathematical Programming*, 91(1):11–31, 2001.

M. L. Fisher. Worst-case analysis of heuristic algorithms. *Management Science*, 26 (1):1 – 17, 1980.

R. J. Francis. *Technology Mapping for Lookup-Table Based Field-Programmable Gate Arrays*. PhD thesis, University of Toronto, 1993.

J. F. Freire Beirão. *Packing Problems in Industrial Environments: Application to the Expedition Problem at INDASA*. PhD thesis, Universidade Técnica de Lisboa, 2009.

D. K. Friesen and M. A. Langston. Variable sized bin packing. *SIAM Journal on Computing*, 15:222–230, 1986.

A. S. Fukunaga and R. E. Korf. Bin completion algorithms for multicontainer packing, knapsack, and covering problems. *Journal of Artificial Intelligence Research*, 28(1):393–429, 2007.

J. Galambos. *The asymptotic theory of extreme order statistics*. John Wiley, New York, 1978.

A. Goel and P. Indyk. Stochastic load balancing and related problems. In *40th Annual Symposium on Foundations of Computer Science*, pages 579–586, 1999.

E. J. Gumbel. *Statistics of Extremes*. Columbia University Press, 1958.

Gurobi Optimization. *Gurobi solver 4.0 reference manual*, 2010.

A. György, G. Lugosi, and G. Ottucsàk. On-line sequential bin packing. *J. Mach. Learn. Res.*, 11:89–109, 2010.

X. Han and K. Makino. Online minimization knapsack problem. In Evripidis Bampis and Klaus Jansen, editors, *Proceedings of the 7th international conference on Approximation and Online Algorithms*, volume 5893 of *WAOA'09*, pages 182 – 193. Springer Berlin Heidelberg, 2010.

X. Han, C. Peng, D. Ye, D. Zhang, and Y. Lan. Dynamic bin packing with unit fraction items revisited. *Inf. Process. Lett.*, 110(23):1049–1054, 2010.

W. Hansen. How accessibility shapes land use. *Journal of the American Institute of Planners*, 25:73–76, 1959.

M. Haouari and M. Serairi. Heuristics for the variable sized bin-packing problem. *Computers & Operations Research*, 36:2877–2884, 2009.

M. Haouari and M. Serairi. Relaxations and exact solution of the variable sized bin packing problem. *Computational Optimization and Applications*, 48:345–368, 2011.

F. Heinicke, A. Simroth, and R. Tadei. On a novel optimisation model and solution method for tactical railway maintenance planning. In *Proceedings of the 2nd International Conference on Road and Rail Infrastructure*, pages 421–427. Department of Transportation, Faculty of Civil Engineering, University of Zagreb, 2012.

F. Heinicke, A. Simroth, R. Tadei, and M. M. Baldi. Job order assignment at optimal costs in railway maintenance. In *Proceedings of the 1st International Conference on Operations Research and Enterprise Systems (ICORES 2013), Barcelona, Spain, 16-18 February, 2013*. SciTePress, 2013. forthcoming.

V. Hemmelmayr, V. Schmid, and C. Blum. Variable neighbourhood search for the variable sized bin packing problem. *Computers & Operations Research*, 39:1097–1108, 2012.

H. Hifi and M. Michrafy. Reduction strategies and exact algorithms for the disjunctively constrained knapsack problem. *Computers & Operations Research*, 34: 2657–2673, 2007.

V. Huang and W. Zhuang. Optimal resource management in packet-switching tdd cdma systems. *IEEE Personal Communications Magazine*, 7:26–31, 2000.

M. D. Hutton. Notes on integer bin-packing for technology mapping on trees. http://www.eecg.toronto.edu/~mdhutton/papers/binpack.pdf, 1993.

ILOG Inc. *IBM ILOG CPLEX v12.1 User's Manual*, 2009.

K. Iwama and S. Taketomi. Removable online knapsack problems. In Peter Widmayer, Stephan Eidenbenz, Francisco Triguero, Rafael Morales, Ricardo Conejo, and Matthew Hennessy, editors, *Automata, Languages and Programming*, volume 2380 of *Lecture Notes in Computer Science*, pages 773 – 773. Springer Berlin / Heidelberg, 2002.

K. Iwama and G. Zhang. Optimal resource augmentations for online knapsack. In *Proceedings of the 10th International Workshop on Approximation and the 11th International Workshop on Randomization, and Combinatorial Optimization. Algorithms and Techniques*, APPROX '07/RANDOM '07, pages 180 – 188. Springer-Verlag, 2007.

K. Iwama and G. Zhang. Online knapsack with resource augmentation. *Information Processing Letters*, 110:1016 – 1020, 2010.

D. S. Johnson. *Near-Optimal bin packing algorithms*. PhD thesis, Dept. of Mathematics, M.I.T., Cambridge, MA, 1973a.

D. S. Johnson. Approximation algorithms for combinatorial problems. In *Proceedings of the fifth annual ACM symposium on Theory of computing*, pages 38–49. ACM, 1973b.

D. S. Johnson. Fast algorithms for bin packing. *Journal of Computer and System Sciences*, 8(3):272–314, 1974.

D. S. Johnson, A. Demeters, J. D. Hullman, M. R. Garey, and R. L. Graham. Worst-case performance bounds for simple one-dimensional packing algorithms. *SIAM Journal on Computing*, 3:299–325, 1974.

J. Kang and S. Park. Algorithms for the variable sized bin packing problem. *European Journal of Operational Research*, 147:365–372, 2003.

N. Karmarkar and R. M. Karp. An efficient approximation scheme for the one-dimensional bin packing problem. In *Proceedings of 23rd Annual Symposium on Foundations of Computer Science*, pages 312–320. IEEE Comput. Soc. Press., 1982.

H. Kellerer, U. Pferschy, and D. Pisinger, editors. *Knapsack Problems*. Springer Verlag, 2004. ISBN 3-540-40286-1.

S. Kosuch and A. Lisser. On two-stage stochastic knapsack problems. *Discrete Applied Mathematics*, doi:10.1016/j.dam.2010.04.006, forthcoming.

B. Kouakou, M. Demange, and E. Soutif. On-line bin-packing problem : maximizing the number of unused bins. Technical report, Université Panthéon-Sorbonne (Paris 1), 2005.

E. L. Lawler and D. E. Wood. Branch-and-bound methods: A survey. *Operations Research*, 14(4):699–719, 1966.

C. C. Lee and D. T. Lee. A simple on-line bin-packing algorithm. *J. ACM*, 32(3): 562–572, 1985.

J. Y.-T Leung and C-L Li. An asymptotic approximation scheme for the concave cost bin packing problem. *European Journal of Operational Research*, 191(2):582 – 586, 2008.

C-L. Li and Z-L. Chen. Bin-packing problem with concave costs of bin utilization. *Na*, 53(4):298–308, 2006.

Z. Li. *Optimal Shipping Decisions in an Airfreight Forwarding Network*. PhD thesis, University of Waterloo, Ontario, Canada, 2011.

G. S. Lueker. Bin packing with items uniformly distributed over intervals [a,b]. In *Proceedings of 24th Annual Symposium on Foundations of Computer Science*, pages 289–297. IEEE Comput. Soc. Press., 1983.

S. Martello and P. Toth. *Knapsack Problems - Algorithms and computer implementations*. John Wiley & Sons, Chichester, UK, 1990.

M. Monaci. *Algorithms for packing and scheduling problems*. PhD thesis, Università di Bologna, Bologna, Italy, 2002.

F. D. Murgolo. An efficient approximation scheme for variable-sized bin packing. *SIAM - Journal on Computing*, 16:149–161, 1987.

D. Naddef and G. Rinaldi. Branch-and-cut algorithms for the capacitated vrp. In Paolo Toth and Daniele Vigo, editors, *The vehicle routing problem*, pages 53–84. Society for Industrial and Applied Mathematics, 2001. ISBN 0-89871-498-2.

J. Peng and B. Zhang. Bin packing problem with uncertain volumes and capacities. http://orsc.edu.cn/online/120601.pdf, 2012.

G. Perboli, R. Tadei, and M. M. Baldi. The stochastic generalized bin packing problem. *Discrete Applied Mathematics*, 160:1291–1297, 2012.

D. Pisinger. *Algorithms for Knapsack Problems*. PhD thesis, University of Copenhagen, Copenhagen, Denmark, 1995.

W. T. Rhee and M. Talagrand. On-line bin packing of items of random size, part i. *SIAM Journal on Computing*, 18:438–445, 1993a.

W. T. Rhee and M. Talagrand. On-line bin packing of items of random size, part ii. *SIAM Journal on Computing*, 18:473–486, 1993b.

K. M. Ross and D. H. K. Tsang. Stochastic knapsack problem. *IEEE Transactions on Communications*, 37:740–747, 1989.

P. Schwerin and G. Wäscher. The bin-packing problem: A problem generator and some numerical experiments with FFD packing and MTP. *International Transactions in Operational Research*, 4:377–389, 1997.

S. Seiden. An optimal online algorithm for bounded space variable-sized bin packing. In Ugo Montanari, JosÃ© Rolim, and Emo Welzl, editors, *Automata, Languages and Programming*, volume 1853 of *Lecture Notes in Computer Science*, pages 283–295. Springer, 2000.

S. Seiden. On the online bin packing problem. *J. ACM*, 49(5):640–671, 2002.

S. Seiden, R. Van Stee, and L. Epstein. New bounds for variable-sized online bin packing. *SIAM Journal on Computing*, 32(2):455–469, 2003.

K. Shintani, A. Imai, E. Nishimura, and S. Papadimitriou. The container shipping network design problem with empty container repositioning. *Transportation Research Part E: Logistics and Transportation Review*, 43:39–59, 2007.

N. Skorin-Kapov. Routing and wavelength assignment in optical networks using bin packing based algorithms. *European Journal of Operational Research*, 177(2): 1167–1179, 2007.

R. Tadei, G. Perboli, and F. Della Croce. A heuristic algorithm for the auto-carrier transportation problem. *Transportation Science*, 36(1):55–62, 2002.

J. D. Ullman. The performance of a memory allocation algorithm. Technical Report 100, Princeton University, 1971.

A. van Vliet. An improved lower bound for on-line bin packing algorithms. *Info Process. Lett.*, 43(5):277–284, 1992.

F. Vanderbeck. Computational study of a column generation algorithm for bin packing and cutting stock problems. *Mathematical Programming*, 86:565–594, 1996.

Z. Wang. Worst-case performance of the successive approximation algorithm for four identical knapsacks. *Journal of Industrial and Management Optimization*, 8 (3):651 – 656, 2012. doi: 10.3934/jimo.2012.8.651.

Z. Wang and W. Xing. A successive approximation algorithm for the multiple knapsack problem. *Journal of Combinatorial Optimization*, 17:347–366, 2009.

A. C. Yao. New algorithms for bin packing. *J. ACM*, 27(2):207–227, 1980.

G Zhang. A new version of on-line variable-sized bin packing. *Discrete Applied Mathematics*, 72(3):193 – 197, 1997.